

[illegible]

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```

LOCK  
V04-

[illegible]



6 3  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 Page 1  
(1)

```
0001 0 MODULE LOCKERS (
0002 0     LANGUAGE (BLISS32),
0003 0     IDENT = 'V04-001',
0004 0     ) =
0005 1 BEGIN
0006 1
0007 1
0008 1 *****
0009 1 *
0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 *  ALL RIGHTS RESERVED
0013 1 *
0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 *  TRANSFERRED.
0020 1 *
0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 *  CORPORATION.
0024 1 *
0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 *
0028 1 *
0029 1 *****
0030 1
0031 1 ++
0032 1
0033 1 FACILITY: F11ACP Structure Level 2
0034 1
0035 1 ABSTRACT:
0036 1
0037 1
0038 1 ENVIRONMENT:
0039 1
0040 1     STARLET operating system, including privileged system services
0041 1     and internal exec routines.
0042 1
0043 1 --
0044 1
0045 1
0046 1 AUTHOR:
0047 1
0048 1 MODIFIED BY:
0049 1
0050 1     V04-001 ACG0463      Andrew C. Goldstein,    6-Sep-1984  15:36
0051 1     Always use volume lock as parent of cache locks
0052 1
0053 1     V03-025 CDS0023      Christian D. Saether    23-Aug-1984
0054 1     Add MAKE_FCB_STALE routine.
0055 1     NEW_ACCESS_LOCK does not need to be global anymore.
0056 1
0057 1     V03-024 CDS0022      Christian D. Saether    15-Aug-1984
```



LOCKERS  
V04-001

H 3  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 Page 2  
(1)

LOCK  
V04-

58	0058	1	Make NEW_ACCESS_LOCK global routine.
59	0059	1	Return zero for lock index in serial_file if
60	0060	1	file number passed in is zero.
61	0061	1	Remove use of MARKDEL in access lock value block.
62	0062	1	Have QEX_N_CANCEL return success if lock actually
63	0063	1	granted, failure if not.
64	0064	1	
65	0065	1	V03-023 CDS0021 Christian D. Saether 14-Aug-1984
66	0066	1	Do not store value block in conv_acclock if it
67	0067	1	was not updated.
68	0068	1	
69	0069	1	V03-022 CDS0020 Christian D. Saether 10-Aug-1984
70	0070	1	Do not attempt to update fcb from value block in
71	0071	1	conv_acclock when the lock is dequeued.
72	0072	1	
73	0073	1	V03-021 CDS0019 Christian D. Saether 6-Aug-1984
74	0074	1	Set STS_HAD_LOCK flag in SERIAL_FILE routine for
75	0075	1	use by OPEN_FILE and CLOSE_FILE.
76	0076	1	
77	0077	1	V03-020 ACG0438 Andrew C. Goldstein, 19-Jul-1984 19:34
78	0078	1	Add CACHE_LOCK subroutine to handle special cache locks;
79	0079	1	add central dequeue routine.
80	0080	1	
81	0081	1	V03-019 CDS0018 Christian D. Saether 18-May-1984
82	0082	1	Force cache miss by incrementing buffer sequence
83	0083	1	numbers on locks when valnotvalid error returned
84	0084	1	by the lock manager.
85	0085	1	
86	0086	1	V03-018 CDS0017 Christian D. Saether 10-May-1984
87	0087	1	Expand test for locking to include whether we are
88	0088	1	a cluster at all.
89	0089	1	
90	0090	1	V03-017 CDS0016 Christian D. Saether 9-May-1984
91	0091	1	Retry on a deadlock error.
92	0092	1	
93	0093	1	V03-016 CDS0015 Christian D. Saether 7-May-1984
94	0094	1	Perform extra conversion on access locks
95	0095	1	after upward conversion to make them sysowned again.
96	0096	1	
97	0097	1	V03-015 CDS0014 Christian D. Saether 5-May-1984
98	0098	1	Bugcheck on access lock conversion failures.
99	0099	1	
100	0100	1	V03-014 CDS0013 Christian D. Saether 20-Apr-1984
101	0101	1	Revise access lock handling.
102	0102	1	Add routines NEW_ACCESS_LOCK, CONV_ACCLOCK,
103	0103	1	ARBITRATE_ACCESS, LOCK_COUNT, QEX_N_CANCEL.
104	0104	1	
105	0105	1	V03-013 CDS0012 Christian D. Saether 10-Apr-1984
106	0106	1	Bump PMS counters for various locks and stalls.
107	0107	1	
108	0108	1	V03-012 ACG0408 Andrew C. Goldstein, 23-Mar-1984 14:46
109	0109	1	Add AST parameter so that impure storage is fully based
110	0110	1	Make APPLY_RVN a macro
111	0111	1	
112	0112	1	V03-011 CDS0011 Christian D. Saether 7-Mar-1984
113	0113	1	Clear BITMAP_VBN when releasing the allocation lock
114	0114	1	so that the Bitscanner doesn't think bitmap_buffer



LOCKERS  
V04-001

I 3  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 Page 3 (1)

```

115      0115      1      is still good.
116      0116      1
117      0117      1      V03-010 CDS0010      Christian D. Saether      19-Feb-1984
118      0118      1      Save/restore VC_SEQNUM in LB_DATASEQ [0].
119      0119      1      Rundown relevant buffers when lowering locks.
120      0120      1      Call WRITE_DIRTY in ALLOCATION_UNLOCK routine.
121      0121      1      Call APPLY_RVN in SERIAL_LOCK for lockbasis.
122      0122      1
123      0123      1      V03-009 CDS0009      Christian D. Saether      30-Dec-1983
124      0124      1      Use L_NORM linkage and BIND_COMMON macro.
125      0125      1
126      0126      1      V03-008 CDS0008      Christian D. Saether      6-Dec-1983
127      0127      1      Use VCB$V_NOSHARE to determine which name space
128      0128      1      a lock is in.
129      0129      1
130      0130      1      V03-007 CDS0007      Christian D. Saether      19-Oct-1983
131      0131      1      Do NOT clone NOALLOC in allocation lock value block.
132      0132      1
133      0133      1      V03-006 CDS0006      Christian D. Saether      15-Oct-1983
134      0134      1      Call FLUSH_LOCK_BASIS from the allocation_unlock
135      0135      1      routine. This fixes a bug in create where the
136      0136      1      allocation lock was being released without invalidating
137      0137      1      the buffer.
138      0138      1
139      0139      1      V03-005 CDS0005      Christian D. Saether      11-Oct-1983
140      0140      1      Add routine to wait for blocking lock and
141      0141      1      rearm if necessary.
142      0142      1
143      0143      1      V03-004 CDS0004      Christian D. Saether      3-Oct-1983
144      0144      1      Set CURR_LCKINDX on serial_file. Clear if match
145      0145      1      on release_serial_lock.
146      0146      1
147      0147      1      V03-003 CDS0003      Christian D. Saether      12-Sep-1983
148      0148      1      New lock name format.
149      0149      1      Save volume context flags.
150      0150      1
151      0151      1      V03-002 CDS0002      Christian D. Saether      28-Jun-1983
152      0152      1      Make access lock a system owned lock.
153      0153      1      Normalize RVN part of lockid for SERIAL_FILE.
154      0154      1
155      0155      1      V03-001 CDS0001      Christian D. Saether      16-May-1983
156      0156      1      SERIAL_FILE resource name does NOT include file sequence
157      0157      1      number.
158      0158      1
159      0159      1      **
160      0160      1
161      0161      1
162      0162      1      LIBRARY 'SYSS$LIBRARY:LIB.L32';
163      0163      1      REQUIRE 'SRC$:FCPDEF.B32';
164      1154      1
165      1155      1      MACRO
166      1156      1      LCK_STS = 0,0,16,0 %;
167      1157      1      LCK_ID = 4,0,32,0 %;
168      1158      1
169      1159      1      FORWARD ROUTINE
170      1160      1      NEW_ACCESS_LOCK : L_NORM,
171      1161      1      LOCK_MODE : L_JSB_1ARG,
```



LOCKERS  
V04-001

J 3  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 Page 4 (1)

:	172	1162	1	CONV_ACCLOCK	:	L_NORM,
:	173	1163	1	ARBITRATE_ACCESS	:	[ JSB_2ARGS,
:	174	1164	1	LOCK_COUNT	:	L_NORM,
:	175	1165	1	QEX_N_CANCEL	:	L_NORM,
:	176	1166	1	MAKE_FCB_STALE	:	L_NORM NOVALUE,
:	177	1167	1	SERIAL_FILE	:	L_NORM,
:	178	1168	1	RELEASE_SERIAL_LOCK	:	L_NORM NOVALUE,
:	179	1169	1	ALLOCATION_LOCK	:	L_NORM NOVALUE,
:	180	1170	1	ALLOCATION_UNLOCK	:	L_NORM NOVALUE,
:	181	1171	1	BLOCK_WAIT	:	L_NORM NOVALUE,
:	182	1172	1	TAKE_BLOCK_LOCK	:	L_NORM NOVALUE,
:	183	1173	1	CACHE_LOCK	:	L_NORM,
:	184	1174	1	DEQ_LOCK	:	L_NORM NOVALUE; ! dequeue a lock

LOCKERS  
V04-001



```
1175 1 ROUTINE NEW_ACCESS_LOCK (LCKMODE, FCB) : L_NORM =
1176 1
1177 1 ++
1178 1
1179 1 FUNCTIONAL DESCRIPTION:
1180 1
1181 1 This routine takes out a lock based on the given lock mode and file id,
1182 1 using the appropriate qualifiers in the resource name.
1183 1
1184 1 CALLING SEQUENCE:
1185 1 See routine header above.
1186 1
1187 1 INPUT PARAMETERS:
1188 1
1189 1 IMPLICIT INPUTS:
1190 1
1191 1 OUTPUT PARAMETERS:
1192 1 NONE
1193 1
1194 1 IMPLICIT OUTPUTS:
1195 1 FCB$$_ACCLCKID - Lock id of granted lock. 0 if no lock granted.
1196 1
1197 1 ROUTINE VALUE:
1198 1 1 if access allowed
1199 1 0 if access not allowed.
1200 1
1201 1 SIDE EFFECTS:
1202 1
1203 1 --
1204 1
1205 2 BEGIN
1206 2
1207 2 BIND_COMMON:
1208 2
1209 2 MAP
1210 2 FCB : REF BBLOCK;
1211 2
1212 2 LOCAL
1213 2 LOCK_BLOCK : BBLOCK [24],
1214 2 RESNAM : VECTOR [24, BYTE],
1215 2 RESNAM_D : VECTOR [2] INITIAL (LONG (22), LONG (RESNAM));
1216 2
1217 2 BIND
1218 2 LOCK_VAL = LOCK_BLOCK + 8 : BBLOCK FIELD (AV);
1219 2
1220 2 EXTERNAL
1221 2 PMSSGL_ACCLCK : ADDRESSING_MODE (ABSOLUTE);
1222 2
1223 2 EXTERNAL ROUTINE
1224 2 XQP$FCBSTALE : ADDRESSING_MODE (ABSOLUTE); ! blocking routine
1225 2
1226 2 ! Generate the resource name to identify the file in the cluster.
1227 2 ! Prefix the entire lock with the facility code for the file system.
1228 2 !
1229 2
1230 2 (RESNAM [0]) = 'F11B';
1231 2 (RESNAM [4])<0,16> = '$a';
```



```
1232 2 CH$MOVE (12,  
1233 2 IF .CURRENT_VCB [VCB$W RVN] EQL 0  
1234 2 THEN CURRENT_VCB [VCB$T_VOLCKNAM]  
1235 2 ELSE CURRENT_RVT [RVT$T_VLSLCKNAM],  
1236 2 RESNAM [6]);  
1237 2  
1238 2  
1239 2 (RESNAM [18]) = .FCB [FCB$L_LOCKBASIS];  
1240 2  
1241 2 PM$SGL_ACCLCK = .PM$SGL_ACCLCK + 1;  
1242 2  
1243 2 !****  
1244 2 ! Attempt to acquire the lock. If granted then access is allowed.  
1245 2 !****  
1246 2  
1247 2 BEGIN  
1248 2 LOCAL  
1249 2 STATUS;  
1250 2  
1251 2 STATUS = $ENQ ( EFN = EFN,  
1252 2 LKMODE = .LKMODE,  
1253 2 FLAGS = LCK$M_NOQUEUE + LCK$M_SYNCSTS + LCK$M_SYSTEM  
1254 2 + LCK$M_VALBLK + LCK$M_NOQUOTA + LCK$M_CVTSYS,  
1255 2 BLKAST = XQP$FCB$STALE,  
1256 2 ASTPRM = .FCB,  
1257 2 LKSB = LOCK_BLOCK,  
1258 2 RESNAM = RESNAM_D);  
1259 2  
1260 2 IF NOT .STATUS  
1261 2 THEN  
1262 2 IF (.STATUS EQL SSS$_NOTQUEUED)  
1263 2 THEN  
1264 2 RETURN 0  
1265 2 ELSE  
1266 2 IF (.STATUS NEQ SSS$_VALNOTVALID)  
1267 2 THEN  
1268 2 ERR_EXIT (.STATUS);  
1269 2  
1270 2 END; ! of block defining STATUS.  
1271 2  
1272 2 FCB [FCB$B_ACCLCKMODE] = .LKMODE;  
1273 2 FCB [FCB$L_ACCLCKID] = .LOCK_BLOCK [LCK_ID];  
1274 2  
1275 2 FCB [FCB$V_DELAYTRNC] = 0;  
1276 2 IF .LOCK_VAL [AV_DELAYTRNC]  
1277 2 THEN  
1278 2 FCB [FCB$V_DELAYTRNC] = 1;  
1279 2 FCB [FCB$L_TRUNCVBVN] = .LOCK_VAL [AV_TRUNCVBVN];  
1280 2  
1281 2 SSS$_NORMAL  
1282 2  
1283 1 END;
```

.TITLE LOCKERS  
.IDENT \V04-001\



.EXTRN PMSSGL ACCLCK, XQPSFCBSTALE  
.EXTRN SYSS\$ENQ

.PSECT \$CODE\$,NOWRT,2

003C 00000 NEW\_ACCESS\_LOCK:

	5E		34	C2	00002	.WORD	Save R2,R3,R4,R5	:	1175
			16	DD	00005	SUBL2	#52, SP	:	
			AE	9E	00007	PUSHL	#22	:	1205
04	AE	08	AE	9E	00007	MOVAB	RESNAM, RESNAM D+4	:	
08	AE	42313146	8F	D0	0000C	MOVL	#1110520134, RESNAM	:	1230
0C	AE	6124	8F	B0	00014	MOVW	#24868, RESNAM+4	:	1231
	50	98	AA	D0	0001A	MOVL	-104(BASE), R0	:	1234
		0E	A0	B5	0001E	TSTW	14(R0)	:	
			07	12	00021	BNEQ	1\$	:	
	50	0080	C0	9E	00023	MOVAB	128(R0), R0	:	1235
			05	11	00028	BRB	2\$	:	1236
OE	50		18	C1	0002A	ADDL3	#24, -100(BASE), R0	:	
AE	9C		0C	28	0002F	MOVC3	#12, (R0), RESNAM+6	:	1237
			AC	D0	00034	MOVL	FCB, R0	:	1239
	1A		A0	D0	00038	MOVL	76(R0), RESNAM+18	:	
		00000000G	9F	D6	0003D	INCL	@PMSSGL_ACCLCK	:	1241
			7E	7C	00043	CLRQ	-(SP)	:	1258
		00000000G	8F	DD	00045	PUSHL	#XQPSFCBSTALE	:	
			50	DD	0004B	PUSHL	R0	:	
			7E	7C	0004D	CLRQ	-(SP)	:	
		18	AE	9F	0004F	PUSHAB	RESNAM D	:	
	7E	7D	8F	9A	00052	MOVZBL	#125, -(SP)	:	
		40	AE	9F	00056	PUSHAB	LOCK_BLOCK	:	
		04	AC	DD	00059	PUSHL	LCKMODE	:	
			1E	DD	0005C	PUSHL	#30	:	
00000000G	00		0B	FB	0005E	CALLS	#11, SYSS\$ENQ	:	
	18		50	E8	00065	BLBS	STATUS, 4\$	:	1260
000009B8	8F		50	D1	00068	CMPL	STATUS, #2488	:	1262
			03	12	0006F	BNEQ	3\$	:	
			50	D4	00071	CLRL	R0	:	1264
			04	00073	RET			:	
000009F0	8F		50	D1	00074	CMPL	STATUS, #2544	:	1266
			03	13	0007B	BEQL	4\$	:	
			50	BF	0007D	CHMU	STATUS	:	1268
			04	0007F	RET			:	
	50	08	AC	D0	00080	MOVL	FCB, R0	:	1272
	0B	04	AC	90	00084	MOVB	LCKMODE, 11(R0)	:	
	50	08	AC	D0	00089	MOVL	FCB, R0	:	1273
48	A0	24	AE	D0	0008D	MOVL	LOCK_BLOCK+4, 72(R0)	:	
	50	08	AC	D0	00092	MOVL	FCB, R0	:	1275
23	A0		02	8A	00096	BICB2	#2, 35(R0)	:	
08	28		01	E1	0009A	BBC	#1, LOCK_VAL, 5\$	:	1276
	50	08	AC	D0	0009F	MOVL	FCB, R0	:	1278
	23		02	88	000A3	BISB2	#2, 35(R0)	:	
	50	08	AC	D0	000A7	MOVL	FCB, R0	:	1279
	50	2C	AE	D0	000AB	MOVL	LOCK_VAL+4, 80(R0)	:	
			01	D0	000B0	MOVL	#1, R0	:	1283
			04	000B3	RET			:	

; Routine Size: 180 bytes, Routine Base: \$CODE\$ + 0000

LOCKERS  
V04-001

: 295

1284 1

N 3  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742 Page 8  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 (2)

LOCKE  
V04-1

IN



```
297 1285 1 GLOBAL ROUTINE LOCK_MODE (ACCTL) : L_JSB_1ARG =
298 1286 1
299 1287 1 !++
300 1288 1
301 1289 1 FUNCTIONAL DESCRIPTION:
302 1290 1
303 1291 1 This routine calculates the appropriate lock mode based on the
304 1292 1 access control flags passed.
305 1293 1
306 1294 1 CALLING SEQUENCE:
307 1295 1 See routine header above.
308 1296 1
309 1297 1 INPUT PARAMETERS:
310 1298 1
311 1299 1 ACCTL - Access control flags
312 1300 1 FIBSV_WRITE - write access requested
313 1301 1 FIBSV_NOREAD - readers disallowed
314 1302 1 FIBSV_NOWRITE - writers disallowed
315 1303 1 FIBSV_NOLOCK - override everyone else's locks
316 1304 1
317 1305 1 IMPLICIT INPUTS:
318 1306 1
319 1307 1 OUTPUT PARAMETERS:
320 1308 1
321 1309 1 IMPLICIT OUTPUTS:
322 1310 1
323 1311 1 ROUTINE VALUE:
324 1312 1
325 1313 1 Lock mode (used as input to SYS$ENQ service).
326 1314 1
327 1315 1 SIDE EFFECTS:
328 1316 1
329 1317 1 --
330 1318 1
331 1319 2 BEGIN
332 1320 2
333 1321 2 MAP
334 1322 2 ACCTL : BBLOCK;
335 1323 2
336 1324 2 BIND_COMMON;
337 1325 2
338 1326 2 ! If NOLOCK is specified and the caller effectively has sysprv on
339 1327 2 the volume, use null lock mode.
340 1328 2
341 1329 2
342 1330 2 IF .ACCTL [FIBSV_NOLOCK] AND .CLEANUP_FLAGS [CLF_SYSPRV]
343 1331 2 THEN
344 1332 2 RETURN LCK$_NLMODE;
345 1333 2
346 1334 2 IF .ACCTL [FIBSV_WRITE]
347 1335 2 THEN
348 1336 2 IF .ACCTL [FIBSV_NOREAD]
349 1337 2 THEN
350 1338 2
351 1339 2 ! This is a writer disallowing readers and writers.
352 1340 2
353 1341 2 LCK$_EXMODE
```



```
1342 2 ELSE
1343 2 IF .ACCTL [FIB$V_NOWRITE]
1344 2 THEN
1345 2
1346 2 ! This is a writer allowing readers but disallowing other writers.
1347 2
1348 2 LCK$K_PWMODE
1349 2 ELSE
1350 2
1351 2 ! This is a writer allowing other readers and writers.
1352 2
1353 2 LCK$K_CWMODE
1354 2 ELSE
1355 2 IF .ACCTL [FIB$V_NOREAD]
1356 2 THEN
1357 2
1358 2 ! This is a reader disallowing other readers and writers.
1359 2
1360 2 LCK$K_EXMODE
1361 2 ELSE
1362 2 IF .ACCTL [FIB$V_NOWRITE]
1363 2 THEN
1364 2
1365 2 ! This is a reader allowing other readers but disallowing writers.
1366 2
1367 2 LCK$K_PRMODE
1368 2 ELSE
1369 2
1370 2 ! This is a reader allowing other readers and writers.
1371 2
1372 2 LCK$K_CRMODE
1373 1 END;
```

07	50	14	E1	00000	LOCK_MODE::			
	03	01	AA	E9	00004	BBC	#20, ACCTL, 1\$	1330
			50	D4	00008	BLBC	1(BASE), 1\$	
				05	0000A	CLRL	R0	1332
OF	50	08	E1	0000B	1\$:	RSB		
OF	50	0A	E0	0000F		BBC	#8, ACCTL, 3\$	1334
	04	50	E9	00013		BBS	#10, ACCTL, 4\$	1336
	50	04	D0	00016		BLBC	ACCTL, 2\$	1343
			05	00019		MOVL	#4, R0	
	50	02	D0	0001A	2\$:	RSB		
			05	0001D		MOVL	#2, R0	1336
04	50	0A	E1	0001E	3\$:	RSB		
	50	05	D0	00022	4\$:	BBC	#10, ACCTL, 5\$	1355
			05	00025		MOVL	#5, R0	
	04	50	E9	00026	5\$:	RSB		
	50	03	D0	00029		BLBC	ACCTL, 6\$	1362
			05	0002C		MOVL	#3, R0	
	50	01	D0	0002D	6\$:	RSB		
			05	00030		MOVL	#1, R0	1373



LOCKERS  
V04-001

D 4  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 Page 11 (3)

; Routine Size: 49 bytes, Routine Base: \$CODES + 00B4

; 386 1374 1

LOCKERS  
V04-001

; Routine



```
388 1375 1 GLOBAL ROUTINE CONV_ACCLOCK (LCKMODEARG, FCBARG) : L_NORM =
389 1376 1
390 1377 1 !++
391 1378 1
392 1379 1 FUNCTIONAL DESCRIPTION:
393 1380 1
394 1381 1 Convert the access lock for the given fcb.
395 1382 1 Dequeue the lock entirely if the ref count is zero.
396 1383 1
397 1384 1 !--
398 1385 1
399 1386 2 BEGIN
400 1387 2
401 1388 2 BIND_COMMON;
402 1389 2
403 1390 2 EXTERNAL ROUTINE
404 1391 2 XQP$FCBSTALE : ADDRESSING_MODE (ABSOLUTE); ! blocking routine
405 1392 2
406 1393 2 LOCAL
407 1394 2 FCB : REF BBLOCK,
408 1395 2 LCKMODE : BYTE,
409 1396 2 LOCK_VAL : REF BBLOCK FIELD (AV),
410 1397 2 LOCK_BLOCK : BBLOCK [24],
411 1398 2 STATUS;
412 1399 2
413 1400 2 FCB = .FCBARG;
414 1401 2 LCKMODE = .LCKMODEARG;
415 1402 2
416 1403 2 ! If lock id field is zero, we are not using locks for this device.
417 1404 2 ! The cluster test is made when the lock is initially acquired.
418 1405 2 !
419 1406 2
420 1407 2 IF (LOCK_BLOCK [LCK_ID] = .FCB [FCB$L_ACCLKID]) EQL 0
421 1408 2 THEN
422 1409 2 RETURN SS$_NORMAL;
423 1410 2
424 1411 2 LOCK_VAL = 0;
425 1412 2
426 1413 2 IF .FCB [FCB$B_ACCLKMODE] GEQU LCK$K_PWMODE
427 1414 2 AND (.FCB [FCB$B_ACCLKMODE] GEQU .LCKMODE)
428 1415 2 THEN
429 1416 2
430 1417 2 ! Store value block on this conversion.
431 1418 2 !
432 1419 2
433 1420 2 BEGIN
434 1421 2 LOCK_VAL = LOCK_BLOCK + 8;
435 1422 2 LOCK_VAL [0,0,32,0] = 0;
436 1423 2 LOCK_VAL [4,0,32,0] = 0;
437 1424 2 LOCK_VAL [8,0,32,0] = 0;
438 1425 2 LOCK_VAL [12,0,32,0] = 0;
439 1426 2
440 1427 2 IF .FCB [FCB$V_DELAYTRNC]
441 1428 2 THEN
442 1429 2 LOCK_VAL [AV_DELAYTRNC] = 1;
443 1430 2
444 1431 2 LOCK_VAL [AV_TRUNCVBVN] = .FCB [FCB$L_TRUNCVBVN];
```



```

445 1432 2 END;
446 1433 2
447 1434 2 IF .FCB [FCB$W_REFCNT] NEQ 0
448 1435 2 THEN
449 1436 2 BEGIN
450 1437 2 LOCAL
451 1438 2 BLOCK_AST;
452 1439 2
453 1440 2 IF .LCKMODE GTRU .FCB [FCB$B_ACCLKMODE]
454 1441 2 THEN
455 1442 2 BLOCK_AST = 0
456 1443 2 ELSE
457 1444 2 BLOCK_AST = XQP$FCBSTALE;
458 1445 2
459 1446 2 STATUS =
460 1447 2 $ENQ ( EFN = EFN,
461 1448 2 LKMODE = .LCKMODE,
462 1449 2 FLAGS = LCK$M_NOQUEUE + LCK$M_SYNCSTS + LCK$M_CONVERT
463 1450 2 + LCK$M_VALBLK + LCK$M_NOQUOTA + LCK$M_CVTSYS,
464 1451 2 BLKAST = .BLOCK_AST,
465 1452 2 ASTPRM = .FCB,
466 1453 2 LKSB = LOCK_BLOCK);
467 1454 2
468 1455 2 IF NOT .STATUS
469 1456 2 THEN
470 1457 2 IF (.STATUS EQL SSS$_NOTQUEUED)
471 1458 2 THEN
472 1459 2 RETURN 0
473 1460 2 ELSE
474 1461 2 IF (.STATUS NEQ SSS$_VALNOTVALID)
475 1462 2 THEN
476 1463 2 BUG_CHECK (XQPERR, 'access lock conversion failure');
477 1464 2
478 1465 2 IF .LCKMODE GTRU .FCB [FCB$B_ACCLKMODE]
479 1466 2 THEN
480 1467 2 IF NOT
481 1468 2 $ENQ ( EFN = EFN,
482 1469 2 LKMODE = .LCKMODE,
483 1470 2 FLAGS = LCK$M_NOQUEUE + LCK$M_SYNCSTS + LCK$M_CONVERT
484 1471 2 + LCK$M_NOQUOTA + LCK$M_CVTSYS,
485 1472 2 BLKAST = XQP$FCBSTALE,
486 1473 2 ASTPRM = .FCB,
487 1474 2 LKSB = LOCK_BLOCK)
488 1475 2 THEN
489 1476 2 BUG_CHECK (XQPERR, 'access lock conversion failure');
490 1477 2
491 1478 2 ! Conversion was successful. Update fcb fields if value block was retrieved.
492 1479 2 ! Store new mode just acquired.
493 1480 2
494 1481 2
495 1482 2 IF .FCB [FCB$B_ACCLKMODE] LSSU LCK$K_PWMODE
496 1483 2 AND (.LCKMODE GEQU .FCB [FCB$B_ACCLKMODE])
497 1484 2 THEN
498 1485 2 BEGIN
499 1486 2 LOCK_VAL = LOCK_BLOCK + 8;
500 1487 2
501 1488 2 FCB [FCB$V_DELAYTRNC] = 0;
```



```
1489 4
1490 4
1491 4
1492 4
1493 4
1494 4
1495 4
1496 4
1497 4
1498 4
1499 4
1500 4
1501 4
1502 4
1503 4
1504 4
1505 4
1506 4
1507 4
1508 4
1509 4
1510 4
1511 4
1512 4
1513 4

IF .LOCK_VAL [AV_DELAYTRNC]
THEN
    FCB [FCB$V_DELAYTRNC] = 1;
    FCB [FCB$L_TRUNCVCBN] = .LOCK_VAL [AV_TRUNCVCBN];
END;

FCB [FCB$B_ACCLKMODE] = .LCKMODE;
END
ELSE
BEGIN
    IF NOT $DEQ (LKID = .LOCK_BLOCK [LCK_ID],
                VALBLK = .LOCK_VAL)
    THEN
        BUG_CHECK (XQPERR, 'Unexpected lock manager error');

    FCB [FCB$L_ACCLKID] = 0;
    FCB [FCB$B_ACCLKMODE] = 0;
END;

SS$NORMAL
END;
```

! of routine CONV\_ACCLOCK

.EXTRN BUG\$\_XQPERR, SYSS\$DEQ

.ENTRY CONV\_ACCLOCK, Save R2,R3,R4,R5,R6

```
MOVAB @#XQPSFCBSTALE, R6
MOVAB SYSS$ENQ, R5
SUBL2 #24, SP
MOVL FCBARG, FCB
MOVB LCKMODEARG, LCKMODE
MOVL 72(FCB), LOCK_BLOCK+4
BNEQ 1$
BRW 14$
CLRL LOCK_VAL
CMPB 11(FCB), #4
BLSSU 3$
CMPB 11(FCB), LCKMODE
BLSSU 3$
MOVAB LOCK_BLOCK+8, LOCK_VAL
CLRQ (LOCK_VAL)
CLRQ 8(LOCK_VAL)
BBC #1, 35(FCB), 2$
BISB2 #2, (LOCK_VAL)
MOVL 80(FCB), 4(LOCK_VAL)
TSTW 24(FCB)
BNEQ 4$
BRW 12$
CMPB LCKMODE, 11(FCB)
BLEQU 5$
CLRL BLOCK_AST
BRB 6$
```

```
1375
1400
1401
1407
1411
1413
1414
1421
1422
1424
1427
1429
1431
1434
1440
1442
```



50		56	D0	0005B	5\$:	MOVL	R6, BLOCK_AST	1444
		7E	7C	0005E	6\$:	CLRQ	-(SP)	1453
		50	DD	00060		PUSHL	BLOCK_AST	
		52	DD	00062		PUSHL	FCB	
		7E	7C	00064		CLRQ	-(SP)	
		7E	D4	00066		CLRL	-(SP)	
7E	6F	8F	9A	00068		MOVZBL	#111, -(SP)	
	20	AE	9F	0006C		PUSHAB	LOCK_BLOCK	
7E		54	9A	0006F		MOVZBL	LCKMODE, -(SP)	
		1E	DD	00072		PUSHL	#30	
65		0B	FB	00074		CALLS	#11, SYSS\$ENQ	
19		50	E8	00077		BLBS	STATUS, 8\$	1455
000009B8	8F	50	D1	0007A		CMPL	STATUS, #2488	1457
		03	12	00081		BNEQ	7\$	
		50	D4	00083		CLRL	R0	1459
			04	00085		RET		
000009F0	8F	50	D1	00086	7\$:	CMPL	STATUS, #2544	1461
		04	13	0008D		BEQL	8\$	
			FEFF	0008F		BUGW		1463
			0000*	00091		.WORD	<BUG\$ XQPERR!4>	
0B	A2	54	91	00093	8\$:	CMPB	LCKMODE, 11(FCB)	1465
		20	1B	00097		BLEQU	9\$	
		7E	7C	00099		CLRQ	-(SP)	1474
		8F	BB	0009B		PUSHR	#*M<R2,R6>	
	0044	7E	7C	0009F		CLRQ	-(SP)	
		7E	D4	000A1		CLRL	-(SP)	
7E	6E	8F	9A	000A3		MOVZBL	#110, -(SP)	
	20	AE	9F	000A7		PUSHAB	LOCK_BLOCK	
7E		54	9A	000AA		MOVZBL	LCKMODE, -(SP)	
		1E	DD	000AD		PUSHL	#30	
65		0B	FB	000AF		CALLS	#11, SYSS\$ENQ	
04		50	E8	000B2		BLBS	R0, 9\$	
			FEFF	000B5		BUGW		1476
			0000*	000B7		.WORD	<BUG\$ XQPERR!4>	
04	0B	A2	91	000B9	9\$:	CMPB	11(FCB), #4	1482
		1B	1E	000BD		BGEQU	11\$	
0B	A2	54	91	000BF		CMPB	LCKMODE, 11(FCB)	1483
		15	1F	000C3		BLSSU	11\$	
53	08	AE	9E	000C5		MOVAB	LOCK_BLOCK+8, LOCK_VAL	1486
23		02	8A	000C9		BICB2	#2, 35(FCB)	1488
04		01	E1	000CD		BBC	#1, (LOCK_VAL), 10\$	1490
23		02	88	000D1		BISB2	#2, 35(FCB)	1492
50		A3	D0	000D5	10\$:	MOVL	4(LOCK_VAL), 80(FCB)	1493
0B		54	90	000DA	11\$:	MOVB	LCKMODE, 11(FCB)	1497
		1B	11	000DE		BRB	14\$	1434
		7E	7C	000E0	12\$:	CLRQ	-(SP)	1503
		53	DD	000E2		PUSHL	LOCK_VAL	
	10	AE	DD	000E4		PUSHL	LOCK_BLOCK+4	
00000000G	00	04	FB	000E7		CALLS	#4, SYSS\$DEQ	
	04	50	E8	000EE		BLBS	R0, 13\$	
			FEFF	000F1		BUGW		1505
			0000*	000F3		.WORD	<BUG\$ XQPERR!4>	
	48	A2	D4	000F5	13\$:	CLRL	72(FCB)	1507
	0B	A2	94	000F8		CLRB	11(FCB)	1508
50		01	D0	000FB	14\$:	MOVL	#1, R0	1513
			04	000FE		RET		

LOCKERS  
V04-001

<sup>1</sup><sub>4</sub>  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 BLISS-32 V4.0-742  
DISK\$VM\$MASTER:[F11X.SRC]LOCKERS.B32;2 Page 16 (4)

; Routine Size: 255 bytes,      Routine Base: \$CODE\$ + 00E5

LOCK  
V04-

; Ro

; 8



```
1514 1 GLOBAL ROUTINE ARBITRATE_ACCESS (ACCTL, FCB) : L_JSB_2ARGS =
1515 1 ++
1516 1 Determine if access to this file is allowed.
1517 1
1518 1 --
1519 1
1520 2 BEGIN
1521 2
1522 2 MAP
1523 2     ACCTL    : BBLOCK,
1524 2     FCB      : REF BBLOCK;
1525 2
1526 2 BIND_COMMON;
1527 2
1528 2 EXTERNAL
1529 2     CLUSGL_CLUB      : ADDRESSING_MODE (GENERAL);
1530 2
1531 2 LOCAL
1532 2     LCKMODE;
1533 2
1534 2 IF .FCB [FCB$W_SEGN] NEQ 0
1535 2 THEN
1536 2     ERR_EXIT (SS$_ACCONFLICT);
1537 2
1538 2 IF NOT (.ACCTL [FIB$V_NOLOCK] AND .CLEANUP_FLAGS [CLF_SYSPRV])
1539 2 THEN
1540 2     IF .FCB [FCB$V_EXCL]
1541 2         OR .ACCTL [FIB$V_NOREAD] AND (.FCB [FCB$W_ACNT] NEQ 0)
1542 2         OR .ACCTL [FIB$V_NOWRITE] AND (.FCB [FCB$W_WCNT] NEQ 0)
1543 2         OR .ACCTL [FIB$V_WRITE] AND (.FCB [FCB$W_LCNT] NEQ 0)
1544 2     THEN
1545 2         ERR_EXIT (SS$_ACCONFLICT);
1546 2
1547 2 IF NOT .BBLOCK [CURRENT_UCB [UCB$L_DEVCHAR2], DEV$V_CLU]
1548 2 OR .CLUSGL_CLUB EQL 0
1549 2 THEN
1550 2     RETURN 1;
1551 2
1552 2 LCKMODE = LOCK_MODE (.ACCTL);
1553 2
1554 2 IF .FCB [FCB$L_ACCLKID] EQL 0
1555 2 THEN
1556 2     NEW_ACCESS_LOCK (.LCKMODE, .FCB)
1557 2 ELSE
1558 2     IF .LCKMODE<0,8> GTRU .FCB [FCB$B_ACCLKMODE]
1559 2     THEN
1560 2         CONV_ACCLOCK (.LCKMODE, .FCB)
1561 2     ELSE
1562 2         SS$_NORMAL
1563 2
1564 1 END;                                ! of routine ARBITRATE_ACCESS
```

.EXTRN CLUSGL\_CLUB

OC BB 00000 ARBITRATE\_ACCESS::

		52		51	D0	00002		PUSHR	#^M<R2,R3>	:	1514
		53		50	D0	00005		MOVL	R1, R2	:	
			2A	A2	B5	00008		MOVL	R0, R3	:	
				27	12	0000B		TSTW	42(FCB)	:	1534
04		53		14	E1	0000D		BNEQ	4\$	:	
		25		AA	E8	00011		BBC	#20, ACCTL, 1\$	:	1538
1A	22	A2	01	03	E0	00015	1\$:	BLBS	1(BASE), 5\$	:	
05		53		0A	E1	0001A		BBS	#3, 34(FCB), 4\$	:	1540
			1A	A2	B5	0001E		BBC	#10, ACCTL, 2\$	:	1541
				11	12	00021		TSTW	26(FCB)	:	
		05		53	E9	00023	2\$:	BNEQ	4\$	:	
			1C	A2	B5	00026		BLBC	ACCTL, 3\$	:	1542
				09	12	00029		TSTW	28(FCB)	:	
0B		53		08	E1	0002B	3\$:	BNEQ	4\$	:	
			1E	A2	B5	0002F		BBC	#8, ACCTL, 5\$	:	1543
				06	13	00032		TSTW	30(FCB)	:	
			0800	8F	BF	00034	4\$:	BEQL	5\$	:	
				3D	11	00038		CHMU	#2048	:	1545
		50		AA	D0	0003A	5\$:	BRB	8\$	:	
		32	94	A0	E9	0003E		MOVL	-108(BASE), R0	:	1547
			3C	00	D5	00042		BLBC	60(R0), 7\$	:	
			00000000G	2A	13	00048		TSTL	CLUSGL_CLUB	:	1548
		50		53	D0	0004A		BEQL	7\$	:	
				FE80	30	0004D		MOVL	ACCTL, R0	:	1552
		53		50	D0	00050		BSBW	LOCK_MODE	:	
			48	A2	D5	00053		MOVL	R0, LCKMODE	:	
				0B	12	00056		TSTL	72(FCB)	:	1554
				52	DD	00058		BNEQ	6\$	:	
				53	DD	0005A		PUSHL	FCB	:	1556
FDBB	CF			02	FB	0005C		PUSHL	LCKMODE	:	
				14	11	00061		CALLS	#2, NEW_ACCESS_LOCK	:	
0B	A2			53	91	00063	6\$:	BRB	8\$	:	
				0B	1B	00067		CMPB	LCKMODE, 11(FCB)	:	1558
				52	DD	00069		BLEQU	7\$	:	
				53	DD	0006B		PUSHL	FCB	:	1560
FE8F	CF			02	FB	0006D		PUSHL	LCKMODE	:	
				03	11	00072		CALLS	#2, CONV_ACCLOCK	:	
		50		01	D0	00074	7\$:	BRB	8\$	:	
				0C	BA	00077	8\$:	MOVL	#1, R0	:	1558
				05	05	00079		POPR	#^M<R2,R3>	:	1564
								RSB		:	

; Routine Size: 122 bytes, Routine Base: \$CODE\$ + 01E4



```
1565 1 GLOBAL ROUTINE LOCK_COUNT (LOCKID) : L_NORM =
1566 1 !++
1567 1 !
1568 1 ! Get count of locks associated with LOCKID lock. Return 1
1569 1 ! if LOCKID is zero.
1570 1 !
1571 1 !--
1572 1
1573 2 BEGIN
1574 2
1575 2 BIND_COMMON;
1576 2
1577 2 LOCAL
1578 2     LOCKCOUNT,
1579 2     STSBLK : VECTOR [4, WORD],
1580 2     ITEMLIST : BBLOCK [12 + 4];
1581 2
1582 2 IF .LOCKID EQL 0
1583 2 THEN
1584 2     RETURN 1;
1585 2
1586 2 ITEMLIST [0,0,16,0] = 4;
1587 2 ITEMLIST [2,0,16,0] = LKIS_LCKCOUNT;
1588 2 ITEMLIST [4,0,32,0] = LOCKCOUNT;
1589 2 ITEMLIST [8,0,32,0] = 0;
1590 2 ITEMLIST [12,0,32,0] = 0;
1591 2
1592 2 IF NOT $GETLKIW (EFN = EFN,
1593 2     LKIDADR = LOCKID,
1594 2     ITMLST = ITEMLIST,
1595 2     IOSB = STSBLK)
1596 2 THEN
1597 2     BUG_CHECK (XQPERR, 'unexpected error from getlki');
1598 2
1599 2 RETURN .LOCKCOUNT
1600 2
1601 1 END;                                ! of routine LOCK_COUNT.
```

```
0000 00000
5E      1C  C2 00002
      04  AC  D5 00005
      04  12 00008
50      01  D0 0000A
      04  00 0000D
04 AE 02050004 8F D0 0000E 1$:
08 AE      6E 9E 00016
      0C  AE 7C 0001A
      7E 7C 0001D
      7E D4 0001F
      20 AE 9F 00021
      14 AE 9F 00024
      04 AC 9F 00027
      1E DD 0002A
```

.EXTRN SYS\$GETLKIW

```
.ENTRY LOCK_COUNT, Save nothing
SUBL2 #28, -SP
TSTL LOCKID
BNEQ 1$
MOVL #1, R0
RET
MOVL #33882116, ITEMLIST
MOVAB LOCKCOUNT, ITEMLIST+4
CLRQ ITEMLIST+8
CLRQ -(SP)
CLRQ -(SP)
CLRL -(SP)
PUSHAB STSBLK
PUSHAB ITEMLIST
PUSHAB LOCKID
PUSHL #30
```

```
: 1565
: 1582
: 1584
: 1586
: 1588
: 1589
: 1595
:
```



000000000G 00  
04

```

00 FB 0002C
50 EB 00033
    FE FF 00036
0000* 00038
6E D0 0003A 2$:
    04 0003D

```

```
CALLS      #7, SYSSGETLKIW
BLBS      R0, 2$
BUGW
.WORD     <BUG$ XQPERR!4>
MOVL      LOCKCOUNT, R0
RET
```

1597  
1599  
1601

```
; Routine Size: 62 bytes,    Routine Base: $CODE$ + 025E
```



```
618 1602 1 GLOBAL ROUTINE QEX_N_CANCEL (LOCKID) : L_NORM =
619 1603 1
620 1604 1 !++
621 1605 1
622 1606 1 Enqueue the given lock for exclusive, cancelling the conversion
623 1607 1 if the lock is not granted.
624 1608 1 !--
625 1609 1
626 1610 1
627 1611 2 BEGIN
628 1612 2
629 1613 2 LOCAL
630 1614 2 LOCK_BLOCK : BBLOCK [8],
631 1615 2 STATUS;
632 1616 2
633 1617 2 IF (LOCK_BLOCK [LCK_ID] = .LOCKID) EQL 0
634 1618 2 THEN
635 1619 2 RETURN 0;
636 1620 2
637 P 1621 2 STATUS = $ENQ (EFN = EFN,
638 P 1622 2 LKMODE = LCK$K_EXMODE,
639 P 1623 2 FLAGS = LCK$M_NOQUOTA + LCK$M_CONVERT + LCK$M_SYNCSTS,
640 1624 2 LKSB = LOCK_BLOCK);
641 1625 2
642 1626 2 IF NOT .STATUS
643 1627 2 THEN
644 1628 2 BUG_CHECK (XQPERR, 'Unexpected lock manager error')
645 1629 2 ELSE
646 1630 2 IF .STATUS EQL SSS_NORMAL ! lock was queued
647 1631 2 THEN
648 1632 2 BEGIN
649 1633 2
650 1634 2 IF NOT $DEQ (LKID = .LOCKID, FLAGS = LCK$M_CANCEL)
651 1635 2 THEN
652 1636 2 BUG_CHECK (XQPERR, 'Unexpected lock manager error');
653 1637 2
654 1638 2 RETURN 0;
655 1639 2 END
656 1640 2 ELSE
657 1641 2 RETURN 1;
658 1642 2
659 1643 1 END;
```

INFO#212 L1:1641  
Null expression appears in value-required context

			0000 00000	.ENTRY	QEX_N_CANCEL, Save nothing	: 1602
	5E		08 C2 00002	SUBL2	#8, -SP	
04	AE	04	AC D0 00005	MOVL	LOCKID, LOCK_BLOCK+4	: 1617
			40 13 0000A	BEQL	3\$	
			7E 7C 0000C	CLRQ	-(SP)	: 1624
			7E 7C 0000E	CLRQ	-(SP)	
			7E 7C 00010	CLRQ	-(SP)	
	7E		2A 7D 00012	MOVQ	#42, -(SP)	



16-Sep-1984 00:38:23 VAX-11 Bliss-32 V4.0-742 Page 22  
 14-Sep-1984 12:30:33 DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32:2 (7)

; Routine Size: 79 bytes, Routine Base: \$CODES + 029C



```
661 1644 1 GLOBAL ROUTINE MAKE_FCB_STALE (FCBARG) : L_NORM NOVALUE =
662 1645 1
663 1646 1 !++
664 1647 1
665 1648 1 FUNCTIONAL DESCRIPTION:
666 1649 1
667 1650 1 For the given FCB, cause the blocking routine to be triggered
668 1651 1 to mark other fcb clusterwide as stale.
669 1652 1
670 1653 1 !--
671 1654 1
672 1655 2 BEGIN
673 1656 2
674 1657 2 BIND_COMMON;
675 1658 2
676 1659 2 EXTERNAL
677 1660 2 CLUSGL_CLUB : ADDRESSING_MODE (GENERAL);
678 1661 2
679 1662 2 LOCAL
680 1663 2 CURRLKMODE : BYTE,
681 1664 2 FCB : REF BBLOCK;
682 1665 2
683 1666 2 IF NOT .BBLOCK [CURRENT_UCB [UCBSL_DEVCHAR2], DEV$V_CLU]
684 1667 2 OR .CLUSGL_CLUB EQL 0
685 1668 2 THEN
686 1669 2 RETURN;
687 1670 2
688 1671 2 FCB = .FCBARG;
689 1672 2
690 1673 2 CURRLKMODE = .FCB [FCBSB_ACCLKMODE];
691 1674 2
692 1675 2 IF .FCB [FCBSL_ACCLKID] EQL 0
693 1676 2 THEN
694 1677 2 IF NOT NEW_ACCESS_LOCK (0, .FCB)
695 1678 2 THEN
696 1679 2 BUG_CHECK (XQPERR, 'unexpected lock manager reaction');
697 1680 2
698 1681 2 IF QEX_N_CANCEL (.FCB [FCBSL_ACCLKID])
699 1682 2 THEN
700 1683 2 ! If the lock was granted, note that by setting the lock mode field in the fcb.
701 1684 2 ! This must be done so that conv_acclock does the correct thing with the value
702 1685 2 ! block.
703 1686 2
704 1687 2
705 1688 2 THEN
706 1689 2 FCB [FCBSB_ACCLKMODE] = LCK$K_EXMODE;
707 1690 2
708 1691 2 CONV_ACCLOCK (.CURRLKMODE, .FCB);
709 1692 2
710 1693 1 END;
```

```
50          94      000C 00000
AA DO 00002
```

```
.ENTRY MAKE_FCB_STALE, Save R2,R3
MOVL -108(BASE), R0
```

```
: 1644
: 1666
```



LOCKERS  
V04-001

D 5  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2  
Page 24  
(8)

3E	3C	A0	E9	00006	BLBC	60(R0), 3\$	:	
	00000000G	00	D5	0000A	TSTL	CLUSGL_CLUB	:	1667
		36	13	00010	BEQL	3\$	:	
52	04	AC	D0	00012	MOVL	FCBARG, FCB	:	1671
53	0B	A2	90	00016	MOVB	11(FCB), CURRLKMODE	:	1673
	48	A2	D5	0001A	TSTL	72(FCB)	:	1675
		10	12	0001D	BNEQ	1\$	:	
		52	DD	0001F	PUSHL	FCB	:	1677
		7E	D4	00021	CLRL	-(SP)	:	
FCED	CF	02	FB	00023	CALLS	#2, NEW_ACCESS_LOCK	:	
	04	50	E8	00028	BLBS	R0, 1\$	:	
			FEFF	0002B	BUGW		:	1679
			0000*	0002D	.WORD	<BUG\$ XQPERR!4>	:	
		48	A2	DD	0002F	PUSHL	:	1681
			01	FB	00032	CALLS	:	
FF7A	CF		50	E9	00037	BLBC	:	
	04		05	90	0003A	MOVB	:	1689
0B	A2		52	DD	0003E	PUSHL	:	1691
			53	9A	00040	MOVZBL	:	
FDB2	7E		02	FB	00043	CALLS	:	
	CF		04	00048	RET		:	1693

; Routine Size: 73 bytes, Routine Base: \$CODE\$ + 02EB



```
712 1694 1 GLOBAL ROUTINE SERIAL_FILE (FID_ADDR) : L_NORM =
713 1695 1
714 1696 1 |++
715 1697 1 |
716 1698 1 | FUNCTIONAL DESCRIPTION:
717 1699 1 |
718 1700 1 | Returns the index of the lock block selected.
719 1701 1 |--
720 1702 1
721 1703 2 BEGIN
722 1704 2
723 1705 2 MAP
724 1706 2 FID_ADDR : REF BBLOCK;
725 1707 2
726 1708 2 BIND_COMMON;
727 1709 2
728 1710 2 EXTERNAL ROUTINE
729 1711 2 CONTINUE_THREAD : L_NORM NOVALUE,
730 1712 2 WAIT_FOR_AST : L_NORM NOVALUE;
731 1713 2
732 1714 2 EXTERNAL
733 1715 2 PM$GL_SYNCHLCK : ADDRESSING_MODE (ABSOLUTE),
734 1716 2 PM$GL_SYNCHWAIT : ADDRESSING_MODE (ABSOLUTE);
735 1717 2
736 1718 2 LOCAL
737 1719 2 RETRIES,
738 1720 2 LB_INDEX,
739 1721 2 LOCK_BASIS : INITIAL (0),
740 1722 2 LOCK_BLOCK : BBLOCK [24],
741 1723 2 RESNAM : VECTOR [10, BYTE],
742 1724 2 RESNAM_D : VECTOR [2] INITIAL (LONG (10), LONG (RESNAM));
743 1725 2
744 1726 2 BIND
745 1727 2 LOCK_VAL = LOCK_BLOCK + 8 : BBLOCK FIELD (FC);
746 1728 2
747 1729 2 LOCK_BASIS<0,16> = .FID_ADDR [FID$W_NUM];
748 1730 2 LOCK_BASIS<16,8> = .FID_ADDR [FID$B_NMX];
749 1731 2
750 1732 2 IF .LOCK_BASIS EQL 0
751 1733 2 THEN
752 1734 2 RETURN 0;
753 1735 2
754 1736 2 LOCK_BASIS<24,8> = .FID_ADDR [FID$B_RVN];
755 1737 2 APPLY_RVN (LOCK_BASIS<24,8>, .CURRENT_RVN);
756 1738 2
757 1739 2 ! Initialize index to 1. 0 is specially reserved for the volume
758 1740 2 ! allocation lock.
759 1741 2 !
760 1742 2
761 1743 2 LB_INDEX = 1;
762 1744 2
763 1745 2 DO
764 1746 2 BEGIN
765 1747 2 IF .LOCK_BASIS EQL .LB_BASIS [.LB_INDEX]
766 1748 2 THEN
767 1749 2 EXITLOOP;
768 1750 2
```

: R

: 1



```

769      1751      3      LB_INDEX = .LB_INDEX + 1;
770      1752      END
771      1753      UNTIL .LB_INDEX EQL LB_NUM;
772      1754
773      1755      IF .LB_INDEX NEQ LB_NUM
774      1756      THEN
775      1757      BEGIN
776      1758      STSFLGS [STS_HAD_LOCK] = 1;
777      1759      CURR_LCKINDX = .LB_INDEX;
778      1760      RETURN .LB_INDEX;
779      1761      END;
780      1762
781      1763      LB_INDEX = 1;
782      1764
783      1765      DO
784      1766      BEGIN
785      1767      IF .LB_LOCKID [.LB_INDEX] EQL 0
786      1768      THEN EXITLOOP;
787      1769      LB_INDEX = .LB_INDEX + 1;
788      1770      END
789      1771      UNTIL .LB_INDEX EQL LB_NUM;
790      1772
791      1773      IF .LB_INDEX EQL LB_NUM
792      1774      THEN
793      1775      BUG_CHECK (XQPERR, 'Failed to find free lock block');
794      1776
795      1777
796      1778      !****
797      1779      ! Generate the resource name to identify the file in the cluster.
798      1780      !****
799      1781
800      1782      ! Prefix the entire lock with the facility code for the file system.
801      1783      !
802      1784
803      1785      (RESNAM [0]) = 'F11B';
804      1786      (RESNAM [4])<0,16> = 's';
805      1787
806      1788      (RESNAM [6]) = .LOCK_BASIS;
807      1789
808      1790      PMSSGL_SYNCHLCK = .PMSSGL_SYNCHLCK + 1;
809      1791
810      1792      RETRIES = 0;
811      1793
812      1794      WHILE 1 DO
813      1795
814      1796      BEGIN
815      1797      LOCAL
816      1798      STATUS;
817      1799
818      P 1800      STATUS = $ENQ ( EFN = EFN,
819      P 1801      LKMODE = LCK$K_PWMODE,
820      P 1802      FLAGS = LCK$M_SYNCSTS + LCK$M_SYSTEM + LCK$M_VALBLK + LCK$M_NOQUOTA,
821      P 1803      LKSB = LOCK_BLOCK,
822      P 1804      PARID = (IF .CURRENT_VCB [VCBSW_RVN] EQL 0
823      P 1805      THEN CURRENT_VCB [VCBSL_VOLLKID]
824      P 1806      ELSE CURRENT_RVT [RVTSL_STRUCLKID]),
825      P 1807      ASTADR = CONTINUE_THREAD,
```



```

826 P 1808          ASTPRM = .BASE,
827       1809          RESNAM = RESNAM_D);
828       1810
829       1811 IF NOT .STATUS
830       1812 THEN
831       1813     BEGIN
832       1814       ERR_EXIT (.STATUS);
833       1815     END;
834       1816
835       1817 IF .STATUS EQL SSS_NORMAL
836       1818 THEN
837       1819     BEGIN
838       1820       PMSSGL_SYNCHWAIT = .PMSSGL_SYNCHWAIT + 1;
839       1821       WAIT_FOR_AST ();
840       1822     END;
841       1823
842       1824 IF NOT .LOCK_BLOCK [LCK_STS]
843       1825 THEN
844       1826     IF .LOCK_BLOCK [LCK_STS] EQL SSS_VALNOTVALID
845       1827     THEN
846       1828
847       1829       ! The value block may not contain the most current information.
848       1830       ! Other nodes may have made more recent modifications that are
849       1831       ! reflected on disk, so we must force a cache miss.
850       1832       ! It may actually be the case that our copy is valid because we were
851       1833       ! the last to touch this, but we can't tell that and the disk
852       1834       ! will have the same contents as our buffer, so the only cost
853       1835       ! is an extra read if that happens.
854       1836
855       1837
856       1838     BEGIN
857       1839       LOCK_VAL [FC_HDRSEQ] = .LOCK_VAL [FC_HDRSEQ] + 1;
858       1840       LOCK_VAL [FC_DATASEQ] = .LOCK_VAL [FC_DATASEQ] + 1;
859       1841       EXITLOOP;
860       1842     END
861       1843   ELSE
862       1844     BEGIN
863       1845       IF .LOCK_BLOCK [LCK_STS] NEQ SSS_DEADLOCK
864       1846       OR .RETRIES GEQ 2
865       1847     THEN
866       1848       ERR_EXIT (.LOCK_BLOCK [LCK_STS]);
867       1849
868       1850       RETRIES = .RETRIES + 1;
869       1851     END
870       1852   ELSE
871       1853     EXITLOOP;    ! i.e., operation was successful
872       1854
873       1855   END;          ! of block defining STATUS.
874       1856
875       1857 LB_LOCKID [.LB_INDEX] = .LOCK_BLOCK [LCK_ID];
876       1858 LB_BASIS [.LB_INDEX] = .LOCK_BASIS;
877       1859
878       1860 LB_HDRSEQ [.LB_INDEX] = .LOCK_VAL [FC_HDRSEQ];
879       1861 LB_DATASEQ [.LB_INDEX] = .LOCK_VAL [FC_DATASEQ];
880       1862
881       1863 CURR_LCKIDX = .LB_INDEX
882       1864
```



			000C 00000		.EXTRN CONTINUE_THREAD	
					.EXTRN WAIT_FOR_AST, PMSSGL_SYNCHLCK	
					.EXTRN PMSSGL_SYNCHWAIT	
					.ENTRY SERIAL_FILE, Save R2,R3	1694
	5E		2C C2 00002		SUBL2 #44, SP	1706
			7E D4 00005		CLRL LOCK_BASIS	
04	AE		0A D0 00007		MOVL #10, RESNAM_D	
08	AE	0C	AE 9E 0000B		MOVAB RESNAM, RESNAM_D+4	
	50	04	AC D0 00010		MOVL FID_ADDR, R0	1729
	6E		60 B0 00014		MOVW (R0), LOCK_BASIS	
02	AE	05	A0 90 00017		MOVB 5(R0), LOCK_BASIS+2	1730
			6E D5 0001C		TSTL LOCK_BASIS	1732
			03 12 0001E		BNEQ 1\$	
			50 D4 00020		CLRL R0	1734
			04 00022		RET	
03	AE	04	A0 90 00023	1\$:	MOVB 4(R0), LOCK_BASIS+3	1736
			05 12 00028		BNEQ 2\$	1737
03	AE	A0	AA 90 0002A		MOVB -96(BASE), LOCK_BASIS+3	
	01	03	AE 91 0002F	2\$:	CMPB LOCK_BASIS+3, #1	
			08 12 00033		BNEQ 3\$	
		A0	AA D5 00035		TSTL -96(BASE)	
			03 12 00038		BNEQ 3\$	
		03	AE 94 0003A		CLRB LOCK_BASIS+3	
	52		01 D0 0003D	3\$:	MOVL #1, LB_INDEX	1743
0080	CA42		6E D1 00040	4\$:	CMPL LOCK_BASIS, 128(BASE)[LB_INDEX]	1747
			07 13 00046		BEQL 5\$	
			52 D6 00048		INCL LB_INDEX	1751
	05		52 D1 0004A		CMPL LB_INDEX, #5	1753
			F1 12 0004D		BNEQ 4\$	
	05		52 D1 0004F	5\$:	CMPL LB_INDEX, #5	1755
			07 13 00052		BEQL 6\$	
A6	AA		02 88 00054		BISB2 #2, -90(BASE)	1758
			00C1 31 00058		BRW 19\$	1759
	52		01 D0 0005B	6\$:	MOVL #1, LB_INDEX	1763
		6C	AA42 D5 0005E	7\$:	TSTL 108(BASE)[LB_INDEX]	1767
			07 13 00062		BEQL 8\$	
			52 D6 00064		INCL LB_INDEX	1769
	05		52 D1 00066		CMPL LB_INDEX, #5	1771
			F3 12 00069		BNEQ 7\$	
	05		52 D1 0006B	8\$:	CMPL LB_INDEX, #5	1773
			04 12 0006E		BNEQ 9\$	
			FEFF 00070		BUGW	1775
			0000* 00072		.WORD <BUG\$ XQPERR!4>	
0C	AE	42313146	8F D0 00074	9\$:	MOVL #1110520134, RESNAM	1785
10	AE	7324	8F B0 0007C		MOVW #29476, RESNAM+4	1786
12	AE		6E D0 00082		MOVL LOCK_BASIS, RESNAM+6	1788
		00000000G	9F D6 00086		INCL @PMSSGL_SYNCHLCK	1790
			53 D4 0008C		CLRL RETRIES	1792
			7E 7C 0008E	10\$:	CLRL -(SP)	1809
			7E D4 00090		CLRL -(SP)	
			5A DD 00092		PUSHL BASE	
		0000G	CF 9F 00094		PUSHAB CONTINUE_THREAD	

50	98	AA	D0	00098	MOVL	-104(BASE), R0	
	0E	A0	B5	0009C	TSTW	14(R0)	
		06	12	0009F	BNEQ	11\$	
50	7C	A0	9E	000A1	MOVAB	124(R0), R0	
		04	11	000A5	BRB	12\$	
50	9C	AA	D0	000A7	11\$: MOVL	-100(BASE), R0	
		60	DD	000AB	12\$: PUSHL	(R0)	
	1C	AE	9F	000AD	PUSHAB	RESNAM_D	
		39	DD	000B0	PUSHL	#57	
	38	AE	9F	000B2	PUSHAB	LOCK_BLOCK	
		04	DD	000B5	PUSHL	#4	
		1E	DD	000B7	PUSHL	#30	
00000000G	00	0B	FB	000B9	CALLS	#11, SYS\$ENQ	
	51	50	D0	000C0	MOVL	R0, STATUS	
	03	51	E8	000C3	BLBS	STATUS, 13\$	1811
		51	BF	000C6	CHMU	STATUS	1814
			04	000C8	RET		
	01	51	D1	000C9	13\$: CMPL	STATUS, #1	1817
		0B	12	000CC	BNEQ	14\$	
		9F	D6	000CE	INCL	@#PMSS\$GL_SYNCHWAIT	1820
0000G	CF	00	FB	000D4	CALLS	#0, WAIT_FOR_AST	1821
	25	18	AE	E8	000D9	14\$: BLBS	LOCK_BLOCK, T8\$
09F0	8F	18	AE	B1	000DD	CMPW	LOCK_BLOCK, #2544
			C3	12	000E3	BNEQ	15\$
		20	AE	D6	000E5	INCL	LOCK_VAL
		24	AE	D6	000E8	INCL	LOCK_VAL+4
			15	11	000EB	BRB	18\$
0E0A	8F	18	AE	B1	000ED	15\$: CMPW	LOCK_BLOCK, #3594
			05	12	000F3	BNEQ	16\$
	02	53	D1	000F5	CMPL	RETRIES, #2	1846
		04	1F	000F8	BLSSU	17\$	
		18	AE	BF	000FA	16\$: CHMU	LOCK_BLOCK
			04	000FD	RET		1848
		53	D6	000FE	17\$: INCL	RETRIES	1850
		8C	11	00100	BRB	10\$	1826
6C AA42	1C	AE	D0	00102	18\$: MOVL	LOCK_BLOCK+4, 108(BASE)[LB_INDEX]	1857
0080 CA42		6E	D0	00108	MOVL	LOCK_BASIS, 128(BASE)[LB_INDEX]	1858
0094 CA42	20	AE	D0	0010E	MOVL	LOCK_VAL, 148(BASE)[LB_INDEX]	1860
00A8 CA42	24	AE	D0	00115	MOVL	LOCK_VAL+4, 168(BASE)[LB_INDEX]	1861
14 AA		52	D0	0011C	19\$: MOVL	LB_INDEX, 20(BASE)	1863
50		52	D0	00120	MOVL	LB_INDEX, R0	1865
			04	00123	RET		

; Routine Size: 292 bytes, Routine Base: \$CODE\$ + 0334

; 884 1866 1



```
1867 1 GLOBAL ROUTINE RELEASE_SERIAL_LOCK (LOCK_INDEX) : L_NORM NOVALUE =
1868 1
1869 1
1870 1 Release the indicated serial_lock.
1871 1
1872 1
1873 2 BEGIN
1874 2
1875 2 EXTERNAL ROUTINE
1876 2     RELEASE_LOCKBASIS : L_NORM,
1877 2     RELEASE_CACHE : L_JSB NOVALUE;
1878 2
1879 2 BIND_COMMON;
1880 2
1881 2 LOCAL
1882 2     LKID_ADDR,
1883 2     LOCK_BLOCK : BBLOCK [24];
1884 2
1885 2 BIND
1886 2     LOCK_VAL = LOCK_BLOCK + 8 : BBLOCK FIELD (FC);
1887 2
1888 2 IF .LOCK_INDEX EQL 0 OR .LOCK_INDEX GEQU LB_NUM
1889 2 THEN
1890 2     BUG_CHECK (XQPERR, 'Invalid lock index');
1891 2
1892 2 LOCK_VAL [FC_HDRSEQ] = .LB_HDRSEQ [.LOCK_INDEX];
1893 2 LOCK_VAL [FC_DATASEQ] = .LB_DATASEQ [.LOCK_INDEX];
1894 2 LOCK_VAL [FC_FILESIZE] = .LB_FILESIZE [.LOCK_INDEX];
1895 2
1896 2 IF (LKID_ADDR = RELEASE_LOCKBASIS (.LOCK_INDEX)) NEQ 0
1897 2 THEN
1898 2     BEGIN
1899 2     LOCAL
1900 2     STATUS;
1901 2
1902 2     LOCK_BLOCK [LCK_ID] = .LB_LOCKID [.LOCK_INDEX];
1903 2
1904 2     STATUS = $ENQ ( EFN = EFN,
1905 2     LKSB = LOCK_BLOCK,
1906 2     FLAGS = LCK$M_CONVERT + LCK$M_VALBLK + LCK$M_CVTSYS + LCK$M_SYNCSTS,
1907 2     LKMODE = LCK$R_NLMODE);
1908 2
1909 2     IF .STATUS<0,16> NEQ SS$ SYNCH
1910 2     OR NOT .LOCK_BLOCK [LCK_STS]
1911 2     THEN
1912 2     BUG_CHECK (XQPERR, 'unexpected lock manager error');
1913 2
1914 2     .LKID_ADDR = .LB_LOCKID [.LOCK_INDEX];
1915 2
1916 2     RELEASE_CACHE ();
1917 2     END
1918 2 ELSE
1919 2     IF NOT $DEQ (LKID = .LB_LOCKID [.LOCK_INDEX],
1920 2     VALBLK = LOCK_VAL)
1921 2     THEN
1922 2     BUG_CHECK (XQPERR, 'Unexpected lock manager error');
1923 2
```

```
1924 2 IF .LOCK_INDEX EQL .CURR_LCKINDX
1925 2 THEN
1926 2     CURR_LCKINDX = 0;
1927 2
1928 2 LB_LOCKID [.LOCK_INDEX] = 0;
1929 2 LB_BASIS [.LOCK_INDEX] = 0;
1930 2
1931 1 END;
```

```

                                000C 00000
                                18 C2 00002
5E                                04 AC D5 00005
                                06 13 00008
                                05 04 AC D1 0000A
                                04 1F 0000E
                                FEFF 00010 1$:
                                0000* 00012
                                50 04 AC D0 00014 2$:
08 AE 0094 CA40 D0 00018
0C AE 00A8 CA40 D0 0001F
10 AE 00BC CA40 D0 00026
                                50 DD 0002D
0000G CF 01 FB 0002F
52 50 D0 00034
                                3F 13 00037
                                50 04 AC D0 00039
04 AE 6C AA40 D0 0003D
                                7E 7C 00043
                                7E 7C 00045
                                7E 7C 00047
                                7E D4 00049
                                7E 4B 8F 9A 0004B
                                20 AE 9F 0004F
                                7E 1E 7D 00052
00000000G 00 0B FB 00055
0689 8F 50 B1 0005C
                                03 12 00061
                                04 6E EB 00063
                                FEFF 00066 3$:
                                0000* 00068
                                50 04 AC D0 0006A 4$:
62 6C AA40 D0 0006E
                                0000G 30 00073
                                1B 11 00076
                                7E 7C 00078 5$:
                                10 AE 9F 0007A
50 04 AC D0 0007D
6C AA40 DD 00081
00000000G 00 04 FB 00085
04 50 EB 0008C
                                FEFF 0008F
                                0000* 00091

.EXTRN RELEASE_LOCKBASIS
.EXTRN RELEASE_CACHE

.ENTRY RELEASE_SERIAL_LOCK, Save R2,R3
SUBL2 #24, SP
TSTL LOCK_INDEX
BEQL 1$
CML PL LOCK_INDEX, #5
BLSSU 2$
BUGW
.WORD <BUG$ XQPERR!4>
MOV L LOCK_INDEX, R0
MOV L 148(BASE)[R0], LOCK_VAL
MOV L 168(BASE)[R0], LOCK_VAL+4
MOV L 188(BASE)[R0], LOCK_VAL+8
PUSHL R0
CALLS #1, RELEASE_LOCKBASIS
MOV L R0, LKID_ADDR
BEQL 5$
MOV L LOCK_INDEX, R0
MOV L 108(BASE)[R0], LOCK_BLOCK+4
CLRQ -(SP)
CLRQ -(SP)
CLRQ -(SP)
CLRQ -(SP)
MOVZBL #75, -(SP)
PUSHAB LOCK_BLOCK
MOVQ #30, -(SP)
CALLS #11, SYSS$ENQ
CMPW STATUS, #1673
BNEQ 3$
BLBS LOCK_BLOCK, 4$
BUGW
.WORD <BUG$ XQPERR!4>
MOV L LOCK_INDEX, R0
MOV L 108(BASE)[R0], (LKID_ADDR)
BSBW RELEASE_CACHE
BRB 6$
CLRQ -(SP)
PUSHAB LOCK_VAL
MOV L LOCK_INDEX, R0
PUSHL 108(BASE)[R0]
CALLS #4, SYSS$DEQ
BLBS R0, 6$
BUGW
.WORD <BUG$ XQPERR!4>
```



16<sup>L</sup>-sep-1984 00:38:23  
14<sup>S</sup>-sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742 Page 32  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 (10)

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```
; Routine Size: 175 bytes,    Routine Base: $CODE$ + 0458
```

: 951 1932 1

LOC  
V04

.....



```

953 1933 1 GLOBAL ROUTINE ALLOCATION_LOCK : L_NORM NOVALUE =
954 1934 1
955 1935 1 |++
956 1936 1 |
957 1937 1 | FUNCTIONAL DESCRIPTION:
958 1938 1 |
959 1939 1 |--
960 1940 1
961 1941 2 BEGIN
962 1942 2
963 1943 2 EXTERNAL
964 1944 2     EX$GL_SYSID_LOCK : ADDRESSING_MODE (GENERAL);
965 1945 2
966 1946 2 BIND_COMMON;
967 1947 2
968 1948 2 EXTERNAL ROUTINE
969 1949 2     CONTINUE_THREAD : L_NORM NOVALUE,
970 1950 2     WAIT_FOR_AST   : L_NORM NOVALUE;
971 1951 2
972 1952 2 EXTERNAL
973 1953 2     PM$GL_VOLLCK   : ADDRESSING_MODE (ABSOLUTE),
974 1954 2     PM$GL_VOLWAIT : ADDRESSING_MODE (ABSOLUTE);
975 1955 2
976 1956 2 LOCAL
977 1957 2     RETRIES,
978 1958 2     LOCK_BLOCK      : BBLOCK [24],
979 1959 2     PARENT_ID,
980 1960 2     RESNAM           : VECTOR [18, BYTE],
981 1961 2     RESNAM_D        : VECTOR [2] INITIAL (LONG (18), LONG (RESNAM));
982 1962 2
983 1963 2 BIND
984 1964 2     LOCK_VAL = LOCK_BLOCK + 8 : BBLOCK FIELD (VC);
985 1965 2
986 1966 2 IF .LB_LOCKID [0] NEQ 0
987 1967 2 THEN
988 1968 2     RETURN;
989 1969 2
990 1970 2 |****
991 1971 2 | Generate the resource name to identify the file in the cluster.
992 1972 2 |****
993 1973 2
994 1974 2 | Prefix the entire lock with the facility code for the file system.
995 1975 2 |
996 1976 2
997 1977 2 (RESNAM [0]) = 'F11B';
998 1978 2 (RESNAM [4])<0,16> = '$v';
999 1979 2
1000 1980 2 CH$MOVE (12, CURRENT_VCB [VCB$T_VOLCKNAM], RESNAM [6]);
1001 1981 2
1002 1982 2 PARENT_ID = 0;
1003 1983 2
1004 1984 2 IF .CURRENT_VCB [VCB$V_NOSHARE]
1005 1985 2 THEN
1006 1986 2     PARENT_ID = .EX$GL_SYSID_LOCK;
1007 1987 2
1008 1988 2 PM$GL_VOLLCK = .PM$GL_VOLLCK + 1;
1009 1989 2
```



```
1010 1990 2 !****
1011 1991 2 ! Attempt to acquire the lock. If granted then access is allowed.
1012 1992 2 !****
1013 1993 2
1014 1994 2 RETRIES = 0;
1015 1995 2
1016 1996 2 WHILE 1 DO
1017 1997 2 BEGIN
1018 1998 2 LOCAL
1019 1999 2     STATUS;
1020 2000 2
1021 2001 2 STATUS = $ENQ ( EFN = EFN,
1022 2002 2     LKMODE = LCK$K_PWMODE,
1023 2003 2     FLAGS = LCK$M_SYNCSTS + LCK$M_SYSTEM + LCK$M_VALBLK + LCK$M_NOQUOTA,
1024 2004 2     PARID = .PARENT_ID,
1025 2005 2     LKSB = LOCK_BLOCK,
1026 2006 2     ASTADR = CONTINUE_THREAD,
1027 2007 2     ASTPRM = .BASE,
1028 2008 2     RESNAM = RESNAM_D);
1029 2009 2
1030 2010 2 IF NOT .STATUS
1031 2011 2 THEN
1032 2012 2     BEGIN
1033 2013 2         ERR_EXIT (.STATUS);
1034 2014 2     END;
1035 2015 2
1036 2016 2 IF .STATUS EQL SS$_NORMAL
1037 2017 2 THEN
1038 2018 2     BEGIN
1039 2019 2         PMSS$GL_VOLWAIT = .PMS$GL_VOLWAIT + 1;
1040 2020 2         WAIT_FOR_AST ();
1041 2021 2     END;
1042 2022 2
1043 2023 2 IF NOT .LOCK_BLOCK [LCK_STS]
1044 2024 2 THEN
1045 2025 2     IF .LOCK_BLOCK [LCK_STS] EQL SS$_VALNOTVALID
1046 2026 2     THEN
1047 2027 2
1048 2028 2 ! Value block may not be the most current. Force a miss on cached
1049 2029 2 ! buffers by incrementing all sequence numbers.
1050 2030 2
1051 2031 2
1052 2032 2     BEGIN
1053 2033 2         (LOCK_VAL [VC_SEQNUM])<0,16> = .(LOCK_VAL [VC_SEQNUM])<0,16> + 1;
1054 2034 2         (LOCK_VAL [VC_SEQNUM])<16,16> = .(LOCK_VAL [VC_SEQNUM])<16,16> + 1;
1055 2035 2         (LOCK_VAL [VC_FLAGS])<1,15> = .(LOCK_VAL [VC_FLAGS])<1,15> + 1;
1056 2036 2     EXITLOOP;
1057 2037 2     END
1058 2038 2 ELSE
1059 2039 2     BEGIN
1060 2040 2         IF .LOCK_BLOCK [LCK_STS] NEQ SS$_DEADLOCK
1061 2041 2         OR .RETRIES GEQ 2
1062 2042 2         THEN
1063 2043 2             ERR_EXIT (.LOCK_BLOCK [LCK_STS]);
1064 2044 2
1065 2045 2     RETRIES = .RETRIES + 1;
1066 2046 2     END
```



```
.EXTRN EXESGL_SYSID_LOCK
.EXTRN PMSSGL_VOLLCK, PMSSGL_VOLWAIT
```

Address	Op Code	Op Name	Comment	Year
007C	00000	ENTRY	ALLOCATION_LOCK, Save R2,R3,R4,R5,R6	1933
5E	30 C2 00002	SUBL2	#48, SP	1944
56	98 AA 9E 00005	MOVAB	-104(BASE), R6	1966
04	08 AE 9E 0000B	PUSHL	#18	1977
6C	AA D5 00010	MOVAB	RESNAM, RESNAM_D+4	1978
01	13 00013	TSTL	108(BASE)	1980
04	00015	BEQL	1\$	1982
08	AE 42313146	RET		1984
0C	AE 7624	MOVL	#1110520134, RESNAM	1986
50	66 D0 00024	MOVW	#30244, RESNAM+4	1988
0080	C0 0C 28 00027	MOVL	(R6), R0	1994
50	53 D4 0002E	MOVW	#12, 128(R0), RESNAM+6	2008
53	AO 66 D0 00030	CLRL	PARENT_ID	
53	00000000G	MOVL	(R6), R0	
00000000G	05 E1 00033	BBC	#5, 83(R0), 2\$	
	00 D0 00038	MOVL	EXESGL SYSID LOCK, PARENT_ID	
	9F D6 0003F	INCL	@#PMSSGL_VOLCK	
	52 D4 00045	CLRL	RETRIES	
	7E 7C 00047	CLRQ	-(SP)	
	7E D4 00049	CLRL	-(SP)	
	5A DD 0004B	PUSHL	BASE	
	0000G	PUSHAB	CONTINUE_THREAD	
	53 DD 00051	PUSHL	PARENT_ID	
	18 AE 9F 00053	PUSHAB	RESNAM_D	
	39 DD 00056	PUSHL	#57	
	3C AE 9F 00058	PUSHAB	LOCK_BLOCK	
	04 DD 0005B	PUSHL	#4	
	1E DD 0005D	PUSHL	#30	
00000000G	00 0B FB 0005F	CALLS	#11, SYS\$ENQ	



			03		50	E8	00066		BLBS	STATUS, 4\$	:	2010
					50	BF	00069		CHMU	STATUS	:	2013
			01			04	0006B		RET		:	
					50	D1	0006C	4\$:	CMPL	STATUS, #1	:	2016
					0B	12	0006F		BNEQ	5\$	:	
					9F	D6	00071		INCL	@#PMSS\$GL_VOLWAIT	:	2019
					00	FB	00077		CALLS	#0, WAIT FOR AST	:	2020
					AE	E8	0007C	5\$:	BLBS	LOCK_BLOCK, 9\$	:	2023
					AE	B1	00080		CMPW	LOCK_BLOCK, #2544	:	2025
					16	12	00086		BNEQ	6\$	:	
					AE	B6	00088		INCW	LOCK_VAL+12	:	2033
					AE	B6	0008B		INCW	LOCK_VAL+14	:	2034
					01	EF	0008E		EXTZV	#1, #15, LOCK_VAL, R0	:	2035
					50	D6	00094		INCL	R0	:	
					50	F0	00096		INSV	R0, #1, #15, LOCK_VAL	:	
					15	11	0009C		BRB	9\$	:	2032
					AE	B1	0009E	6\$:	CMPW	LOCK_BLOCK, #3594	:	2040
					05	12	000A4		BNEQ	7\$	:	
					52	D1	000A6		CMPL	RETRIES, #2	:	2041
					04	1F	000A9		BLSSU	8\$	:	
					AE	BF	000AB	7\$:	CHMU	LOCK_BLOCK	:	2043
						04	000AE		RET		:	
					52	D6	000AF	8\$:	INCL	RETRIES	:	2045
					94	11	000B1		BRB	3\$	:	2025
					01	CE	000B3	9\$:	MNEGL	#1, 128(BASE)	:	2052
					AE	D0	000B8		MOVL	LOCK_BLOCK+4, 108(BASE)	:	2053
					AE	D0	000BD		MOVL	LOCK_VAL+12, 168(BASE)	:	2054
					AE	B0	000C3		MOVW	LOCK_VAL, -92(BASE)	:	2059
					66	D0	000C8		MOVL	(R6), R0	:	2060
					AE	90	000CB		MOVB	LOCK_VAL+2, 58(R0)	:	
					66	D0	000D0		MOVL	(R6), R0	:	2061
					AE	90	000D3		MOVB	LOCK_VAL+3, 59(R0)	:	
					66	D0	000D8		MOVL	(R6), R0	:	2062
					AE	D0	000DB		MOVL	LOCK_VAL+4, 64(R0)	:	
					B6	D0	000E0		MOVL	@0(R6), R0	:	2063
					AE	D0	000E4		MOVL	LOCK_VAL+8, 60(R0)	:	
					04	12	000E9		BNEQ	10\$	:	2064
					FEFF		000EB		BUGW		:	2066
					0000*		000ED		.WORD	<BUG\$_XQPERR!4>	:	
					04		000EF	10\$:	RET		:	2068

; Routine Size: 240 bytes, Routine Base: \$CODE\$ + 0507

; 1089 2069 1



```
1091 2070 1 GLOBAL ROUTINE ALLOCATION_UNLOCK : L_NORM NOVALUE =
1092 2071 1
1093 2072 1 ! This routine updates the value block and releases the current
1094 2073 1 ! volume allocation lock, if any.
1095 2074 1
1096 2075 1
1097 2076 2 BEGIN
1098 2077 2
1099 2078 2 BIND_COMMON;
1100 2079 2
1101 2080 2 LOCAL
1102 2081 2     LOCK_BLOCK      : BBLOCK [24],
1103 2082 2     LOCKID;
1104 2083 2
1105 2084 2 EXTERNAL ROUTINE
1106 2085 2     WRITE_DIRTY      : L_NORM,
1107 2086 2     RELEASE_LOCKBASIS : L_NORM;
1108 2087 2
1109 2088 2 BIND
1110 2089 2     LOCK_VAL = LOCK_BLOCK + 8 : BBLOCK FIELD (VC);
1111 2090 2
1112 2091 2 IF (LOCKID = .LB_LOCKID [0]) EQL 0
1113 2092 2 THEN
1114 2093 2     RETURN;
1115 2094 2
1116 2095 2 WRITE_DIRTY (-1);
1117 2096 2
1118 2097 2 RELEASE_LOCKBASIS (0);
1119 2098 2
1120 2099 2 BITMAP_VBN = 0;
1121 2100 2
1122 2101 2 LOCK_VAL [VC_FLAGS] = .SAVE VC FLAGS;
1123 2102 2 LOCK_VAL [VC_IBMAPVBN] = .CURRENT_VCB [VCBSB_IBMAPVBN];
1124 2103 2 LOCK_VAL [VC_SBMAPVBN] = .CURRENT_VCB [VCBSB_SBMAPVBN];
1125 2104 2 LOCK_VAL [VC_VOLFREE] = .CURRENT_VCB [VCBSL_FREE];
1126 2105 2 LOCK_VAL [VC_IDXFILEOF] = .BBLOCK [.CURRENT_VCB [VCBSL_FCBFL], FCB$E_FBLK];
1127 2106 2 LOCK_VAL [VC_SEQNUM] = .LB_DATASEQ [0];
1128 2107 2
1129 2108 2 LB_LOCKID [0] = 0;
1130 2109 2 LB_BASIS [0] = 0;
1131 2110 2
1132 P 2111 2 IF NOT $DEQ (LKID = .LOCKID,
1133 2112 2     VALBLK = LOCK_VAL)
1134 2113 2 THEN
1135 2114 2     BUG_CHECK (XQPERR, 'Unexpected lock manager error');
1136 2115 2
1137 2116 1 END;
! of routine ALLOCATION_UNLOCK
```

.EXTRN WRITE\_DIRTY

```
SE      0004 00000
52      18 C2 00002
        6C AA D0 00005
        5D 13 00009
7E      01 CE 0000B
```

```
.ENTRY ALLOCATION_UNLOCK, Save R2
SUBL2 #24, SP
MOVL 108(BASE), LOCKID
BEQL 1$
MNEGL #1, -(SP)
```

```
: 2070
:
: 2091
:
: 2095
```



LOCKERS  
V04-001

E 6  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 (12)

Page 38

0000G	CF		01	FB	0000E	CALLS	#1, WRITE_DIRTY	:	
			7E	D4	00013	CLRL	-(SP)	:	2097
0000G	CF		01	FB	00015	CALLS	#1, RELEASE_LOCKBASIS	:	
		B4	AA	D4	0001A	CLRL	-76(BASE)	:	2099
08	AE	A4	AA	B0	0001D	MOVW	-92(BASE), LOCK_VAL	:	2101
	50	98	AA	D0	00022	MOVL	-104(BASE), R0	:	2102
0A	AE	3A	AO	90	00026	MOVB	58(R0), LOCK_VAL+2	:	
	50	98	AA	D0	0002B	MOVL	-104(BASE), R0	:	2103
0B	AE	3B	AO	90	0002F	MOVB	59(R0), LOCK_VAL+3	:	
	50	98	AA	D0	00034	MOVL	-104(BASE), R0	:	2104
0C	AE	40	AO	D0	00038	MOVL	64(R0), LOCK_VAL+4	:	
	50	98	BA	D0	0003D	MOVL	@-104(BASE), R0	:	2105
10	AE	3C	AO	D0	00041	MOVL	60(R0), LOCK_VAL+8	:	
14	AE	00A8	CA	D0	00046	MOVL	168(BASE), LOCK_VAL+12	:	2106
		6C	AA	D4	0004C	CLRL	108(BASE)	:	2108
		0080	CA	D4	0004F	CLRL	128(BASE)	:	2109
			7E	7C	00053	CLRQ	-(SP)	:	2112
		10	AE	9F	00055	PUSHAB	LOCK_VAL	:	
			52	DD	00058	PUSHL	LOCKID	:	
00000000G	00		04	FB	0005A	CALLS	#4, SYSSDEQ	:	
	04		50	E8	00061	BLBS	R0, 1\$	:	
				FEFF	00064	BUGW		:	2114
				0000*	00066	.WORD	<BUG\$_XQPERR!4>	:	
				04	00068	RET		:	2116

; Routine Size: 105 bytes, Routine Base: \$CODE\$ + 05F7

; 1138 2117 1



```
1140 2118 1 GLOBAL ROUTINE BLOCK_WAIT : L_NORM NOVALUE =
1141 2119 1
1142 2120 2 BEGIN
1143 2121 2
1144 2122 2 BUILTIN
1145 2123 2 TESTBITSS;
1146 2124 2
1147 2125 2 LOCAL
1148 2126 2 ACB : REF BBLOCK,
1149 2127 2 BLOCKID_A,
1150 2128 2 ACTIVITY_A,
1151 2129 2 RESNAM : VECTOR [20, BYTE],
1152 2130 2 RESNAM_D : VECTOR [2] INITIAL (LONG (18), LONG (RESNAM)),
1153 2131 2 LOCK_BLOCK : BBLOCK [8],
1154 2132 2 STATUS;
1155 2133 2
1156 2134 2 EXTERNAL
1157 2135 2 SCH$GL_SWPPID : ADDRESSING_MODE (GENERAL),
1158 2136 2 XQ$DEBLOCKER : ADDRESSING_MODE (GENERAL);
1159 2137 2
1160 2138 2 BIND_COMMON;
1161 2139 2
1162 2140 2 EXTERNAL ROUTINE
1163 2141 2 CONTINUE_THREAD : L_NORM NOVALUE,
1164 2142 2 XQ$BLOCK_ROUTINE : ADDRESSING_MODE (GENERAL),
1165 2143 2 WAIT_FOR_AST : L_NORM NOVALUE;
1166 2144 2
1167 2145 2 (RESNAM [0]) = 'F11B';
1168 2146 2 (RESNAM [4])<0,16> = 'sb';
1169 2147 2
1170 2148 2 IF .CURRENT_VCB [VCB$W_RVN] EQL 0
1171 2149 2 THEN
1172 2150 2 BEGIN
1173 2151 2 ACB = CURRENT_VCB [VCB$B_ACB];
1174 2152 2 ACTIVITY_A = CURRENT_VCB [VCB$W_ACTIVITY];
1175 2153 2 BLOCKID_A = CURRENT_VCB [VCB$L_BLOCKID];
1176 2154 2 CH$MOVE (12, CURRENT_VCB [VCB$T_VOLCKNAM], RESNAM [6]);
1177 2155 2 END
1178 2156 2 ELSE
1179 2157 2 BEGIN
1180 2158 2 ACB = CURRENT_RVT [RVT$B_ACB];
1181 2159 2 ACTIVITY_A = CURRENT_RVT [RVT$W_ACTIVITY];
1182 2160 2 BLOCKID_A = CURRENT_RVT [RVT$L_BLOCKID];
1183 2161 2 CH$MOVE (12, CURRENT_RVT [RVT$T_VLSLCKNAM], RESNAM [6]);
1184 2162 2 END;
1185 2163 2
1186 P 2164 2 STATUS = $ENQ ( EFN = EFN,
1187 P 2165 2 LKMODE = LCK$K_PWMODE,
1188 P 2166 2 FLAGS = LCK$M_SYNCSTS + LCK$M_SYSTEM + LCK$M_NOQUOTA,
1189 P 2167 2 LKSB = LOCK_BLOCK,
1190 2168 2 ASTADR = CONTINUE_THREAD,
1191 P 2169 2 ASTPRM = .BASE,
1192 2170 2 RESNAM = RESNAM_D);
1193 2171 2
1194 2172 2 IF NOT .STATUS
1195 2173 2 THEN
1196 2174 2 ERR_EXIT (.STATUS);
```



```
1197 2175 2 IF .STATUS EQL SSS_NORMAL
1198 2176 2 THEN
1199 2177 2     WAIT_FOR_AST ();
1200 2178 2
1201 2179 2 STATUS = .LOCK_BLOCK [LCK_STS];
1202 2180 2
1203 2181 2 IF NOT .STATUS
1204 2182 2 THEN
1205 2183 2     ERR_EXIT (.STATUS);
1206 2184 2
1207 2185 2 IF ..BLOCKID_A NEQ 0
1208 2186 2 THEN
1209 2187 2     BEGIN
1210 2188 2         DEQ_LOCK (.LOCK_BLOCK [LCK_ID]);
1211 2189 2         RETURN
1212 2190 2     END;
1213 2191 2
1214 2192 2 ! Because we have a PW lock on the blocking lock, AND the lock id is
1215 2193 2 ! zero, it is now our duty to rearm the blocking lock. Note that
1216 2194 2 ! another blocking lock (EX) may already be queued, and the following
1217 2195 2 ! conversion to CR with the sysblock routine will immediately fire such
1218 2196 2 ! that when we return from the ENQ activity will be disabled again.
1219 2197 2 ! That all works ok because we are going to retest conditions once we
1220 2198 2 ! exit from this routine, and will simply re-execute it.
1221 2199 2
1222 2200 2
1223 2201 2 IF TESTBITSS (.ACTIVITY_A)
1224 2202 2 THEN
1225 2203 2     BUG_CHECK (XQPERR, 'unexpected activity flag state');
1226 2204 2
1227 2205 2 ACB [ACBSB_RMOD] = PSL$C_KERNEL + ACBSM_NODELETE;
1228 2206 2 ACB [ACBSL_PID] = .SCH$G[SWPPID];
1229 2207 2 ACB [ACBSL_AST] = XQP$DEQBLOKER;
1230 2208 2 ACB [ACBSL_ASTPRM] = .LOCK_BLOCK [LCK_ID];
1231 2209 2
1232 2210 2 .BLOCKID_A = .LOCK_BLOCK [LCK_ID];
1233 2211 2
1234 2212 2 STATUS = $ENQ ( EFN = EFN,
1235 2213 2                 LKMODE = LCK$K_CRMODE,
1236 2214 2                 FLAGS = LCK$M_CVTSYS + LCK$M_CONVERT + LCK$M_SYNCSTS
1237 2215 2                     + LCK$M_NOQUOTA + LCK$M_NOQUEUE,
1238 2216 2                 LKSB = LOCK_BLOCK,
1239 2217 2                 BLKAST = XQP$BLOCK_ROUTINE,
1240 2218 2                 ASTPRM = .CURRENT_VCB);
1241 2219 2
1242 2220 2 IF .STATUS
1243 2221 2 THEN
1244 2222 2     STATUS = .LOCK_BLOCK [LCK_STS];
1245 2223 2
1246 2224 2 IF NOT .STATUS
1247 2225 2 THEN
1248 2226 2     BUG_CHECK (XQPERR, 'Unexpected lock manager reaction');
1249 2227 2
1250 2228 2
1251 2229 2 1 END;
```



				OBFC 00000	.EXTRN SCH\$GL_SWPPID, XQPSDEQBLOCKER		
					.EXTRN XQPSBLOCK_ROUTINE		
					.ENTRY BLOCK_WAIT, Save R2,R3,R4,R5,R6,R7,R8,R9,-	2118	
		5B	00000000G	00 9E 00002	MOVAB	SYSENG, R11	
		5E		24 C2 00009	SUBL2	#36, SP	
	08	AE		12 D0 0000C	MOVL	#18, RESNAM_D	2120
	0C	AE	10	AE 9E 00010	MOVAB	RESNAM, RESNAM_D+4	
		59	98	AA 9E 00015	MOVAB	-104(BASE), R9	2136
	10	AE	42313146	8F D0 00019	MOVL	#1110520134, RESNAM	2145
	14	AE	6224	8F B0 00021	MOVW	#25124, RESNAM+4	2146
		56		69 D0 00027	MOVL	(R9), R6	2148
			0E	A6 B5 0002A	TSTW	14(R6)	
				21 12 0002D	BNEQ	1\$	
		56	00AB	C6 9E 0002F	MOVAB	168(R6), ACB	2151
	57	69	000000A0	8F C1 00034	ADDL3	#160, (R9), ACTIVITY_A	2152
	58	69	0000008C	8F C1 0003C	ADDL3	#140, (R9), BLOCKID_A	2153
		50		69 D0 00044	MOVL	(R9), R0	2154
16	AE	0080	C0	0C 28 00047	MOVW	#12, 128(R0), RESNAM+6	
				19 11 0004E	BRB	2\$	2148
	56	9C	AA	28 C1 00050	ADDL3	#40, -100(BASE), ACB	2158
	57	9C	AA	06 C1 00055	ADDL3	#6, -100(BASE), ACTIVITY_A	2159
	58	9C	AA	24 C1 0005A	ADDL3	#36, -100(BASE), BLOCKID_A	2160
			9C	AA D0 0005F	MOVL	-100(BASE), R0	2161
16	AE	18	A0	0C 28 00063	MOVW	#12, 24(R0), RESNAM+6	
				7E 7C 00069	CLRQ	-(SP)	2170
				7E D4 0006B	CLRL	-(SP)	
				5A DD 0006D	PUSHL	BASE	
			0000G	CF 9F 0006F	PUSHAB	CONTINUE_THREAD	
			20	7E D4 00073	CLRL	-(SP)	
			20	AE 9F 00075	PUSHAB	RESNAM_D	
				38 DD 00078	PUSHL	#56	
				AE 9F 0007A	PUSHAB	LOCK_BLOCK	
				04 DD 0007D	PUSHL	#4	
				1E DD 0007F	PUSHL	#30	
		68		0B FB 00081	CALLS	#11, SYSENG	
		52		50 D0 00084	MOVL	R0, STATUS	
		10		52 E9 00087	BLBC	STATUS, 4\$	2172
		01		52 D1 0008A	CMPL	STATUS, #1	2176
				05 12 0008D	BNEQ	3\$	
		0000G	CF	00 FB 0008F	CALLS	#0, WAIT FOR AST	2178
			52	6E 3C 00094	MOVZWL	LOCK_BLOCK, STATUS	2180
			03	52 E8 00097	BLBS	STATUS, 5\$	2182
				52 BF 0009A	CHMU	STATUS	2184
				04 0009C	RET		
				68 D5 0009D	TSTL	(BLOCKID_A)	2186
				09 13 0009F	BEQL	6\$	
		0000V	CF	AE DD 000A1	PUSHL	LOCK_BLOCK+4	2189
				01 FB 000A4	CALLS	#1, DEQ_LOCK	
				04 000A9	RET		2188
04		67		00 E3 000AA	BBCS	#0, (ACTIVITY_A), 7\$	2202
				FEFF 000AE	BUGW		2204
				0000* 000B0	.WORD	<BUG\$_XQPERR!4>	
	0B	A6		20 90 000B2	MOVW	#32, T1(ACB)	2206
	0C	A6	00000000G	00 D0 000B6	MOVL	SCH\$GL_SWPPID, 12(ACB)	2207



10	A6	00000000G	00	9E	000BE	MOVAB	XQPSDEQBLOCKER, 16(ACB)	:	2208
14	A6	04	AE	D0	000C6	MOVL	LOCK_BLOCK+4, 20(ACB)	:	2209
	68	04	AE	D0	000CB	MOVL	LOCK_BLOCK+4, (BLOCKID_A)	:	2211
			7E	7C	000CF	CLRQ	-(SP)	:	2219
		00000000G	00	9F	000D1	PUSHAB	XQPSBLOCK_ROUTINE	:	
			69	DD	000D7	PUSHL	(R9)	:	
			7E	7C	000D9	CLRQ	-(SP)	:	
			7E	D4	000DB	CLRL	-(SP)	:	
	7E	6E	8F	9A	000DD	MOVZBL	#110, -(SP)	:	
		20	AE	9F	000E1	PUSHAB	LOCK_BLOCK	:	
			01	DD	000E4	PUSHL	#1	:	
			1E	DD	000E6	PUSHL	#30	:	
	6B		0B	FB	000E8	CALLS	#11, SYS\$ENQ	:	
	52		50	D0	000EB	MOVL	R0, STATUS	:	
	06		52	E9	000EE	BLBC	STATUS, 8\$	:	2221
	52		6E	3C	000F1	MOVZWL	LOCK_BLOCK, STATUS	:	2223
	04		52	E8	000F4	BLBS	STATUS, 9\$	:	2225
				FEFF	000F7	BUGW		:	2227
				0000*	000F9	.WORD	<BUG\$_XQPERR!4>	:	
				04	000FB	RET		:	2229

; Routine Size: 252 bytes, Routine Base: \$CODE\$ + 0660

; 1252 2230 1



```
1254 2231 1 GLOBAL ROUTINE TAKE_BLOCK_LOCK : L_NORM NOVALUE =
1255 2232 1
1256 2233 2 BEGIN
1257 2234 2
1258 2235 2 LOCAL
1259 2236 2     RESNAM      : VECTOR [20, BYTE],
1260 2237 2     RESNAM_D    : VECTOR [2] INITIAL (LONG (18), LONG (RESNAM)),
1261 2238 2     STATUS,
1262 2239 2     LOCK_BLOCK  : BBLOCK [8];
1263 2240 2
1264 2241 2 BIND_COMMON;
1265 2242 2
1266 2243 2 EXTERNAL ROUTINE
1267 2244 2     CONTINUE_THREAD : L_NORM,
1268 2245 2     WAIT_FOR_AST   : L_NORM;
1269 2246 2
1270 2247 2 (RESNAM [0]) = 'F11B';
1271 2248 2 (RESNAM [4]) < 0, 16 > = 'sb';
1272 2249 2
1273 2250 2 IF .CURRENT_VCB [VCB$W_RVN] EQL 0
1274 2251 2 THEN
1275 2252 2     BEGIN
1276 2253 2     CH$MOVE (12, CURRENT_VCB [VCB$T_VOLCKNAM], RESNAM [6]);
1277 2254 2     END
1278 2255 2 ELSE
1279 2256 2     BEGIN
1280 2257 2     CH$MOVE (12, CURRENT_RVT [RVT$T_VLSLCKNAM], RESNAM [6]);
1281 2258 2     END;
1282 2259 2
1283 P 2260 2 STATUS = $ENQ ( EFN = EFN,
1284 P 2261 2     LKMODE = LCK$K_EXMODE,
1285 P 2262 2     FLAGS = LCK$M_SYNCSTS + LCK$M_SYSTEM + LCK$M_NOQUOTA,
1286 P 2263 2     LKSB = LOCK_BLOCK,
1287 P 2264 2     ASTADR = CONTINUE_THREAD,
1288 2265 2     ASTPRM = .BASE,
1289 2266 2     RESNAM = RESNAM_D);
1290 2267 2
1291 2268 2 IF NOT .STATUS
1292 2269 2 THEN
1293 2270 2     ERR_EXIT (.STATUS);
1294 2271 2
1295 2272 2 IF .STATUS EQL SSS_NORMAL
1296 2273 2 THEN
1297 2274 2     WAIT_FOR_AST ();
1298 2275 2
1299 2276 2 STATUS = .LOCK_BLOCK [LCK_STS];
1300 2277 2
1301 2278 2 IF NOT .STATUS
1302 2279 2 THEN
1303 2280 2     ERR_EXIT (.STATUS);
1304 2281 2
1305 2282 2 BLOCK_LOCKID = .LOCK_BLOCK [LCK_ID];
1306 2283 2
1307 2284 2 END;
```



					003C 00000	.ENTRY	TAKE_BLOCK_LOCK, Save R2,R3,R4,R5	:	2231
					24 C2 00002	SUBL2	#36, SP	:	
	08	5E			12 D0 00005	MOVL	#18, RESNAM_D	:	2233
	0C	AE			AE 9E 00009	MOVAB	RESNAM, RESNAM_D+4	:	
	10	AE	42313146		8F D0 0000E	MOVL	#1110520134, RESNAM	:	2247
	14	AE	6224		8F B0 00016	MOVW	#25124, RESNAM+4	:	2248
		50	98		AA D0 0001C	MOVL	-104(BASE), R0	:	2250
			0E		A0 B5 00020	TSTW	14(R0)	:	
					09 12 00023	BNEQ	1\$	:	
16	AE	0080	C0		0C 28 00025	MOV3	#12, 128(R0), RESNAM+6	:	2253
					0A 11 0002C	BRB	2\$	:	2250
16	AE	18	50	9C	AA D0 0002E	MOVL	-100(BASE), R0	:	2257
			A0		0C 28 00032	MOV3	#12, 24(R0), RESNAM+6	:	
					7E 7C 00038	CLRQ	-(SP)	:	2266
					7E D4 0003A	CLRL	-(SP)	:	
					5A DD 0003C	PUSHL	BASE	:	
			0000G		CF 9F 0003E	PUSHAB	CONTINUE_THREAD	:	
					7E D4 00042	CLRL	-(SP)	:	
			20		AE 9F 00044	PUSHAB	RESNAM_D	:	
					38 DD 00047	PUSHL	#56	:	
			20		AE 9F 00049	PUSHAB	LOCK_BLOCK	:	
					05 DD 0004C	PUSHL	#5	:	
					1E DD 0004E	PUSHL	#30	:	
			00000000G	00	0B FB 00050	CALLS	#11, SYS\$ENQ	:	
				52	50 D0 00057	MOVL	R0, STATUS	:	
				10	52 E9 0005A	BLBC	STATUS, 4\$	:	2268
				01	52 D1 0005D	CMPL	STATUS, #1	:	2272
					05 12 00060	BNEQ	3\$	:	
			0000G	CF	00 FB 00062	CALLS	#0, WAIT_FOR_AST	:	2274
				52	6E 3C 00067	MOVZWL	LOCK_BLOCK, STATUS	:	2276
				03	52 E8 0006A	BLBS	STATUS, 5\$	:	2278
					52 BF 0006D	CHMU	STATUS	:	2280
					04 0006F	RET		:	
	FF7C	CA	04	AE	D0 00070	MOVL	LOCK_BLOCK+4, -132(BASE)	:	2282
					04 00076	RET		:	2284

; Routine Size: 119 bytes, Routine Base: \$CODE\$ + 075C



```
1309 2285 1 GLOBAL ROUTINE CACHE_LOCK (LOCK_BASIS, LOCK_ID, MODE) : L_NORM =
1310 2286 1
1311 2287 1 ++
1312 2288 1
1313 2289 1 FUNCTIONAL DESCRIPTION:
1314 2290 1
1315 2291 1 This routine takes out the specified special purpose cache
1316 2292 1 lock identified by the lock basis.
1317 2293 1
1318 2294 1 CALLING SEQUENCE:
1319 2295 1 CACHE_LOCK (LOCK_BASIS, LOCK_ID, MODE)
1320 2296 1
1321 2297 1 INPUT PARAMETERS:
1322 2298 1 LOCK_BASIS: basis for lock name
1323 2299 1 LOCK_ID: address of existing lock ID or 0
1324 2300 1 MODE: mode of operation:
1325 2301 1 0 = normal cache lock, system, noqueue
1326 2302 1 1 = force flush lock, process, queued
1327 2303 1 2 = write access lock, system, queued
1328 2304 1
1329 2305 1 IMPLICIT INPUTS:
1330 2306 1 CURRENT_VCB: VCB of volume
1331 2307 1 CURRENT_RVT: RVT of volume set
1332 2308 1 XQP$GL_FILESERVER: PID of file server process
1333 2309 1
1334 2310 1 OUTPUT PARAMETERS:
1335 2311 1 LOCK_ID: address to store ID of lock taken
1336 2312 1
1337 2313 1 IMPLICIT OUTPUTS:
1338 2314 1 NONE
1339 2315 1
1340 2316 1 ROUTINE VALUE:
1341 2317 1 LBC if lock not available
1342 2318 1 LBS if lock successfully taken
1343 2319 1
1344 2320 1 SIDE EFFECTS:
1345 2321 1 NONE
1346 2322 1
1347 2323 1 --
1348 2324 1
1349 2325 2 BEGIN
1350 2326 2
1351 2327 2 LOCAL
1352 2328 2 STATUS, system service status
1353 2329 2 LOCK_STATUS : BBLOCK [8], lock status block
1354 2330 2 RESNAM : VECTOR [10, BYTE], ! resource name string
1355 2331 2 RESNAM_D : VECTOR [2], resource name string descriptor
1356 2332 2 PARENT_ID, parent ID for lock
1357 2333 2 LOCK_MODE, mode of lock to take
1358 2334 2 LOCK_FLAGS : BBLOCK [4], $ENQ mode flags
1359 2335 2 CACHE_TYPE; type code of cache being locked
1360 2336 2
1361 2337 2 EXTERNAL
1362 2338 2 XQP$GL_FILESERVER : ADDRESSING_MODE (GENERAL);
1363 2339 2 ! PID of cache server process
1364 2340 2
1365 2341 2 EXTERNAL ROUTINE
```



```

1366      2342      2      CONTINUE_THREAD : L_NORM,      ! continue after wait
1367      2343      2      WAIT_FOR_AST : L_NORM,      ! wait for completion AST
1368      2344      2      XQPS$ONLOCK_CACHE : [L_NORM ADDRESSING_MODE (GENERAL);
1369      2345      2      ! system cache blocking AST routine
1370      2346      2
1371      2347      2      BIND_COMMON;
1372      2348      2
1373      2349      2      ! Set up resource name and parent ID. Get the existing lock ID if any.
1374      2350      2      !
1375      2351      2
1376      2352      2      (RESNAM[0]) = 'F11B';
1377      2353      2      (RESNAM[4]) <0,16> = 'sc';
1378      2354      2      (RESNAM[6]) = LOCK_BASIS;
1379      2355      2      RESNAM_D[0] = 10;
1380      2356      2      RESNAM_D[1] = RESNAM;
1381      2357      2
1382      2358      2      PARENT_ID = .CURRENT_VCB[VCB$$_VOLLKID];
1383      2359      2
1384      2360      2      LOCK_MODE = LCK$$_CWMODE;
1385      2361      2
1386      2362      2      LOCK_FLAGS = LCK$$_SYNCSYS OR LCK$$_SYSTEM OR LCK$$_NOQUOTA;
1387      2363      2      IF (LOCK_STATUS[LOCK_ID] = ..LOCK_ID) NEQ 0
1388      2364      2      THEN LOCK_FLAGS = LCK$$_CONVERT;
1389      2365      2
1390      2366      2      ! Check that the file server process is running. If not, we cannot take
1391      2367      2      ! out the normal cache lock because no one can respond to the blocking
1392      2368      2      ! AST. For the normal cache lock, use PR mode, noqueue.
1393      2369      2      !
1394      2370      2
1395      2371      2      IF .MODE EQL 0
1396      2372      2      THEN
1397      2373      2      BEGIN
1398      2374      2      IF .XQPS$$_FILESERVER EQL 0
1399      2375      2      THEN RETURN 0;
1400      2376      2      LOCK_MODE = LCK$$_PRMODE;
1401      2377      2      LOCK_FLAGS = .LOCK_FLAGS OR LCK$$_NOQUEUE;
1402      2378      2      END;
1403      2379      2
1404      2380      2      ! First take out the lock in process mode.
1405      2381      2      !
1406      2382      2
1407      2383      2
1408      2384      2      STATUS = $ENQ (EFN = EFN,
1409      2385      2      LKMODE = .LOCK_MODE,
1410      2386      2      FLAGS = .LOCK_FLAGS,
1411      2387      2      LKSB = LOCK_STATUS,
1412      2388      2      RESNAM = RESNAM_D,
1413      2389      2      PARID = .PARENT_ID,
1414      2390      2      ASTADR = CONTINUE_THREAD,
1415      2391      2      ASTPRM = .BASE
1416      2392      2      );
1417      2393      2      IF NOT .STATUS
1418      2394      2      AND .STATUS NEQ $$$_NOTQUEUED
1419      2395      2      THEN
1420      2396      2      IF .LOCK_FLAGS[LCK$$_CONVERT]
1421      2397      2      THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
1422      2398      2      ELSE ERR_EXIT (.STATUS);
```



```

: 1423      2399 2
: 1424      2400 2 IF .STATUS EQL SSS NORMAL
: 1425      2401 2 THEN WAIT_FOR_AST (?);
: 1426      2402 2
: 1427      2403 2 IF .STATUS
: 1428      2404 2 THEN
: 1429      2405 2 BEGIN
: 1430      2406 2     STATUS = .LOCK_STATUS[LCK_STS];
: 1431      2407 2     IF NOT .STATUS
: 1432      2408 2     THEN
: 1433      2409 2         IF .LOCK_FLAGS[LCK$V CONVERT]
: 1434      2410 2         THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
: 1435      2411 2         ELSE ERR_EXIT (.STATUS);
: 1436      2412 2     LOCK_FLAGS = .LOCK_FLAGS OR LCK$M_CONVERT;
: 1437      2413 2
: 1438      2414 2 ! If the lock needs to be system owned (normal cache lock and write
: 1439      2415 2 ! access lock) now convert it.
: 1440      2416 2
: 1441      2417 2
: 1442      2418 2 IF NOT .MODE
: 1443      2419 2 THEN
: 1444      2420 2 BEGIN
: 1445      2421 2     LOCK_FLAGS = .LOCK_FLAGS OR LCK$M_CVTSYS;
: 1446      2422 2     CACHE_TYPE = .LOCK_BASIS<0,24>;
: 1447      2423 2     IF .CACHE_TYPE GTD FIB$C_EXTENT CACHE
: 1448      2424 2     THEN CACHE_TYPE = FIB$C_QUOTA_CACHE;
: 1449      2425 2     STATUS = $ENQ (EFN = EFN,
: 1450      2426 2         LKMODE = .LOCK_MODE,
: 1451      2427 2         FLAGS = .LOCK_FLAGS,
: 1452      2428 2         LKSB = LOCK_STATUS,
: 1453      2429 2         RESNAM = RESNAM D,
: 1454      2430 2         PARID = .PARENT_ID,
: 1455      2431 2         BLKAST = XQP$UNLOCK_CACHE,
: 1456      2432 2         ASTPRM = .CURRENT_UCB OR .CACHE_TYPE
: 1457      2433 2     );
: 1458      2434 2 IF NOT .STATUS
: 1459      2435 2 THEN
: 1460      2436 2     IF .LOCK_FLAGS[LCK$V CONVERT]
: 1461      2437 2     THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error')
: 1462      2438 2     ELSE ERR_EXIT (.STATUS);
: 1463      2439 2 END;
: 1464      2440 2 END;
: 1465      2441 2
: 1466      2442 2 IF .STATUS
: 1467      2443 2 THEN .LOCK_ID = .LOCK_STATUS[LCK_ID];
: 1468      2444 2
: 1469      2445 2 .STATUS
: 1470      2446 1 END;

```

! End of routine CACHE\_LOCK

```

.EXTRN XQP$GL_FILESERVER
.EXTRN XQP$UNLOCK_CACHE

```

```

.ENTRY CACHE_LOCK, Save R2,R3,R4,R5,R6
MOVAB SYS$ENQ, R6
SUBL2 #24, SP

```

```

56 00000000G 007C 00000
5E          00 9E 00002
          18 C2 00009

```

```

: 2285
:
:

```



04	AE	42313146	8F	D0	0000C	MOVL	#1110520134, RESNAM	2353
08	AE	6324	8F	B0	00014	MOVW	#25380, RESNAM+4	2354
0A	AE	04	AC	D0	0001A	MOVL	LOCK_BASIS, RESNAM+6	2355
			0A	DD	0001F	PUSHL	#10	2356
04	AE	08	AE	9E	00021	MOVAB	RESNAM, RESNAM_D+4	2357
50		98	AA	D0	00026	MOVL	-104(BASE), R0	2359
55		7C	A0	D0	0002A	MOVL	124(R0), PARENT_ID	
54			02	D0	0002E	MOVL	#2, LOCK_MODE	2361
53			38	D0	00031	MOVL	#56, LOCK_FLAGS	2363
18	AE	08	BC	D0	00034	MOVL	@LOCK_ID, LOCK_STATUS+4	2364
			03	13	00039	BEQL	1\$	
53			02	D0	0003B	MOVL	#2, LOCK_FLAGS	2365
		0C	AC	D5	0003E	TSTL	MODE	2372
			11	12	00041	BNEQ	3\$	
		00000000G	00	D5	00043	TSTL	XQP\$GL_FILESERVER	2375
			03	12	00049	BNEQ	2\$	
			50	D4	0004B	CLRL	R0	2376
				04	0004D	RET		
54			03	D0	0004E	MOVL	#3, LOCK_MODE	2377
53			04	88	00051	BISB2	#4, LOCK_FLAGS	2378
			7E	7C	00054	CLRQ	-(SP)	2392
			7E	D4	00056	CLRL	-(SP)	
			5A	DD	00058	PUSHL	BASE	
		0000G	CF	9F	0005A	PUSHAB	CONTINUE_THREAD	
			55	DD	0005E	PUSHL	PARENT_ID	
		18	AE	9F	00060	PUSHAB	RESNAM_D	
			53	DD	00063	PUSHL	LOCK_FLAGS	
		34	AE	9F	00065	PUSHAB	LOCK_STATUS	
			54	DD	00068	PUSHL	LOCK_MODE	
			1E	DD	0006A	PUSHL	#30	
		66	0B	FB	0006C	CALLS	#11, SYS\$ENQ	
		52	50	D0	0006F	MOVL	R0, STATUS	
		11	52	E8	00072	BLBS	STATUS, 4\$	2393
	000009B8		8F	D1	00075	CMPL	STATUS, #2488	2394
			52	D1	00075	CMPL	STATUS, #2488	
			08	13	0007C	BEQL	4\$	
69			01	E1	0007E	BBC	#1, LOCK_FLAGS, 8\$	2396
				FEFF	00082	BUGW		2397
				0000*	00084	.WORD	<BUG\$ XQPERR!4>	
			52	D1	00086	CMPL	STATUS, #1	2400
			05	12	00089	BNEQ	5\$	
		0000G	00	FB	0008B	CALLS	#0, WAIT_FOR_AST	2401
			52	E9	00090	BLBC	STATUS, T1\$	2403
			52	3C	00093	MOVZWL	LOCK_STATUS, STATUS	2406
		14	52	E8	00097	BLBS	STATUS, 6\$	2407
			01	E1	0009A	BBC	#1, LOCK_FLAGS, 8\$	2409
4D				FEFF	0009E	BUGW		2410
				0000*	000A0	.WORD	<BUG\$ XQPERR!4>	
			02	88	000A2	BISB2	#2, LOCK_FLAGS	2412
			AC	E8	000A5	BLBS	MODE, 9\$	2418
		0C	8F	88	000A9	BISB2	#64, LOCK_FLAGS	2421
		40	00	EF	000AD	EXTZV	#0, #24, LOCK_BASIS, CACHE_TYPE	2422
50	04	AC	50	D1	000B3	CMPL	CACHE_TYPE, #2	2423
			03	1B	000B6	BLEQU	7\$	
			03	D0	000B8	MOVL	#3, CACHE_TYPE	2424
			7E	7C	000BB	CLRQ	-(SP)	2433
		00000000G	00	9F	000BD	PUSHAB	XQP\$UNLOCK_CACHE	
7E	94	AA	50	C9	000C3	BISL3	CACHE_TYPE, -108(BASE), -(SP)	



			7E D4 000C8	CLRL	-(SP)	
			55 DD 000CA	PUSHL	PARENT_ID	
		18	AE 9F 000CC	PUSHAB	RESNAM_D	
			53 DD 000CF	PUSHL	LOCK_FLAGS	
		34	AE 9F 000D1	PUSHAB	LOCK_STATUS	
			54 DD 000D4	PUSHL	LOCK_MODE	
			1E DD 000D6	PUSHL	#30	
	66		0B FB 000D8	CALLS	#11, SYSENG	
	52		50 D0 000DB	MOVL	R0, STATUS	
	10		52 E8 000DE	BLBS	STATUS, 10\$	2434
06	53		01 E1 000E1	BBC	#1, LOCK_FLAGS, 8\$	2436
			FEFF 000E5	BUGW		2437
			0000* 000E7	.WORD	<BUG\$_XQPERR!4>	
			03 11 000E9	BRB	9\$	
			52 BF 000EB 8\$:	CHMU	STATUS	2438
			04 000ED	RET		
	05		52 E9 000EE 9\$:	BLBC	STATUS, 11\$	2442
08	BC	18	AE D0 000F1 10\$:	MOVL	LOCK_STATUS+4, @LOCK_ID	2443
	50		52 D0 000F6 11\$:	MOVL	STATUS, R0	2446
			04 000F9	RET		

; Routine Size: 250 bytes, Routine Base: \$CODE\$ + 07D3



```
: 1472 2447 1 GLOBAL ROUTINE DEQ_LOCK (LOCK_ID) : L_NORM NOVALUE =
: 1473 2448 1
: 1474 2449 1 ++
: 1475 2450 1
: 1476 2451 1 FUNCTIONAL DESCRIPTION:
: 1477 2452 1
: 1478 2453 1 This is a central routine for doing simple dequeues of locks.
: 1479 2454 1 It bugchecks on error.
: 1480 2455 1
: 1481 2456 1 CALLING SEQUENCE:
: 1482 2457 1 DEQ_LOCK (LOCK_ID)
: 1483 2458 1
: 1484 2459 1 INPUT PARAMETERS:
: 1485 2460 1 LOCK_ID: ID of lock to release
: 1486 2461 1
: 1487 2462 1 IMPLICIT INPUTS:
: 1488 2463 1 NONE
: 1489 2464 1
: 1490 2465 1 OUTPUT PARAMETERS:
: 1491 2466 1 NONE
: 1492 2467 1
: 1493 2468 1 IMPLICIT OUTPUTS:
: 1494 2469 1 NONE
: 1495 2470 1
: 1496 2471 1 ROUTINE VALUE:
: 1497 2472 1 NONE
: 1498 2473 1
: 1499 2474 1 SIDE EFFECTS:
: 1500 2475 1 NONE
: 1501 2476 1
: 1502 2477 1 --
: 1503 2478 1
: 1504 2479 2 BEGIN
: 1505 2480 2
: 1506 2481 3 IF NOT $DEQ (LKID = .LOCK_ID)
: 1507 2482 2 THEN BUG_CHECK (XQPERR, FATAL, 'Unexpected lock manager error');
: 1508 2483 2
: 1509 2484 1 END; ! End of routine DEQ_LOCK
```

		0000 0000	.ENTRY	DEQ_LOCK, Save nothing	: 2447
		7E 7C 00002	CLRQ	-(SP)	: 2481
		7E D4 00004	CLRL	-(SP)	
		AC DD 00006	PUSHL	LOCK_ID	
		04 FB 00009	CALLS	#4, SYSSDEQ	
		50 E8 00010	BLBS	R0, 1\$	
		FEFF 00013	BUGW		: 2482
		0000* 00015	.WORD	<BUG\$_XQPERR!4>	
		04 00017 1\$:	RET		: 2484

; Routine Size: 24 bytes, Routine Base: \$CODE\$ + 08CD

; 1510 2485 1



LOCKERS  
V04-001

E 7  
16-Sep-1984 00:38:23  
14-Sep-1984 12:30:33

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[F11X.SRC]LOCKERS.B32;2 (16)

Page 51

: 1511 2486 1 END  
: 1512 2487 0 ELUDOM

PSECT SUMMARY

: Name Bytes Attributes  
: \$CODE\$ 2277 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

: File Total Symbols Loaded Percent Pages Mapped Processing Time  
: \_\$255\$DUA28:[SYSLIB]LIB.L32;1 18619 89 0 1000 00:01.9

: Information: 1  
: Warnings: 0  
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:LOCKERS/OBJ=OBJ\$:LOCKERS MSRC\$:LOCKERS/UPDATE=(ENH\$:LOCKERS)

: Size: 2277 code + 0 data bytes  
: Run Time: 02:15.8  
: Elapsed Time: 04:19.9  
: Lines/CPU Min: 1099  
: Lexemes/CPU-Min: 66753  
: Memory Used: 281 pages  
: Compilation Complete

MAK  
V04



0171 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

